SANDIA REPORT

SAND2015-0862
Unlimited Release
Printed February 2015

Nested Narratives Final Report

Andrew T. Wilson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

SAND2015-0862
Unlimited Release
Printed February 2015

Nested Narratives Final Report

Andrew T. Wilson
Scalable Analysis and Visualization
Sandia National Laboratories
P.O. Box 5800
MS 1326
Albuquerque, NM 87185-1326
atwilso@sandia.gov

Bradley J. Carvey
Scalable Analysis and Visualization
Sandia National Laboratories
P.O. Box 5800
MS 1326
Albuquerque, NM 87185-1326
bjcarve@sandia.gov

J. Christopher Forsythe
Human Factors
Sandia National Laboratories
P.O. Box 5800
MS 0830
Albuquerque, NM 87185-0830
jcforsy@sandia.gov

Nicholas D. Pattengale
Critical Systems Security
Sandia National Laboratories
P.O. Box 5800
MS 0672
Albuquerque, NM 87185-0672
ndpatte@sandia.gov

Abstract

In cybersecurity forensics and incident response, the story of what has happened is the most important artifact yet the one least supported by tools and techniques. Existing tools focus on gathering and manipulating low-level data to allow an analyst to investigate exactly what happened on a host system or a network. Higher-level analysis is usually left to whatever ad hoc tools and techniques an individual may have developed.

We discuss visual representations of narrative in the context of cybersecurity incidents with an eye toward multi-scale illustration of actions and actors. We envision that this representation could smoothly encompass individual packets on a wire at the lowest level and nation-state-level actors at the highest. We present progress to date, discuss the impact of technical risk on this project and highlight opportunities for future work.

Acknowledgment

We are deeply grateful to the staff and management of Sandia's Cyber Incident Response department for their assistance and warm welcome over the course of this project.

Contents

Appendix

List of Figures

List of Tables

Chapter 1

Introduction

We begin with the customary statement of the obvious: the impact and consequence of decisions made in cybersecurity incident response is difficult to overestimate. That makes it critically important to get the message right when conveying information from the front lines, where incident responders deal with bits, bytes and packets, to those who will use the message as a basis for further action.

In light of this, we have alarmingly little support for constructing narratives of who did what to whom in a cyber context. An informal survey of a list of 125 cybersecurity tools includes 60 for vulnerability assessment, 17 for monitoring and 10 for forensics. Of those ten, only 6 are specific to cybersecurity and all six focus on fine-grained detail instead of action and motivation. This leaves analysts to construct the bigger picture without any assistance or direct connection to the data. The artifacts they create (often sets of PowerPoint slides) are then handed up the line, culled, transformed and summarized further, introducing the risk that the final story that a decision-maker hears may be distorted and completely disconnected from the original information.

Nested Narratives was a Sandia National Laboratories Laboratory Directed Research and Development (LDRD) project that aimed to bridge this gap. Our original intent had three foci:

1. Attribute traffic on a network to the processes and user actions that caused it
2. Help analysts construct stories from data that tell not only *what* is happening but also *why*
3. Preserve those stories in multi-layered artifacts that can be used to tell the story at any level from strategic intent down to actual data

In this report we describe the work done under the auspices of Nested Narratives. We report our original intent as well as what actually happened and the reasons why.

1.1 Our Goals

Our ultimate goal is to strengthen analysts in their unique and irreplaceable craft: condense fact, detail and story from the wilderness of raw data. We aim first to help an analyst assemble the

narrative itself; second, to present it to audiences using varying levels of technical and strategic detail.

The first challenge in assembling a story is to filter an avalanche of raw data down to just the pertinent subset. This is currently a manual and incomplete process. We build on Pattengale's LAASER project to instrument the Linux kernel to associate running processes with the network traffic they generate. This nascent capability allows automatic extraction of a causal chain of network traffic related to events of interest on a targeted computer – events that we already know how to identify.

Our second challenge is to represent actors, actions, evidence and intent as elements of a narrative. Such a representation combines elements of a relationship graph, a timeline and a storyboard. It must accommodate bridges between events disparate in space and time, different time scales (milliseconds to months), segmentation of events into units of meaning and assembly of these units into hierarchy. It must also support different interpretations and remember its own history for later reflection.

Our third challenge is to make it easy and fluid for humans to build and present these narrative artifacts. Just as most representations focus on one kind of entity, most existing presentation tools focus on just that kind of entity. Powerpoint works with images and bulleted lists. Analyst's Notebook [12] is an analysis tool that helps its users marshal evidence into separate association charts and timelines. Storyboards capture action, view and sequence while leaving context and relationship information to other media. Our task here is to use aspects of each of these representations where appropriate.

Our fourth challenge is to measure the impact of our research. Cybersecurity analysts and domain experts already have tools to automate the detection of anomalies on networks. We will assess whether our prototype tools help them construct a big picture from these anomalies. Our hypothesis is that tools that augment the cognitive processes associated with narrative construction will result in better operational performance. We planned to measure this by comparing the performance of a control group using existing tools and an experimental group asked to use new tools.

Chapter 2

Related Work

We begin by distinguishing *visualization of narrative* from *visual storytelling*. The latter begins with cave paintings and includes nearly every recorded non-textual form of communication invented since then: illustration, graphic novels, photography, cinematography, and serial comics are just a few examples. Visual storytelling depicts *sequences of actions taken by characters in a setting*. Conversely, we define *visualization of narrative* to mean an illustration of *the structure of a story*: which characters interact at what times, where those interactions take place, precedence between events and how individual events fit together to tell larger stories. In Nested Narratives we focus on visualization of narrative structure.

2.1 Visualizing Co-Location

A large class of narrative visualization methods focuses on illustrating *co-location*: situations where two or more characters (as individuals or as groups) interact with the environment and with one another. Most of these methods were inspired by an issue of the online comic strip `xkcd` entitled Movie Charts [18]. Movie Charts comprises hand-drawn visualizations of characters, groups and their interactions from five different stories told in popular movies. [1] These charts show character introductions and exits, major events, and the formation and dissolution of groups as the story progresses.

Subsequent efforts have focused on software to build such diagrams by hand with algorithmically assisted layout (PlotWeaver [22]) or purely automatically using metadata about character co-location [13].. Ogawa and Ma have applied a similar rendering technique [20] to the evolution of source code in large software projects.

[1] The tangled lines shown at bottom right in Figure 2.1 are for the movie Primer. Its plot involves time travel and features main characters interacting with themselves at different points along their respective timelines. Proper discussion of this movie requires topological knot theory as well as several new verb tenses.

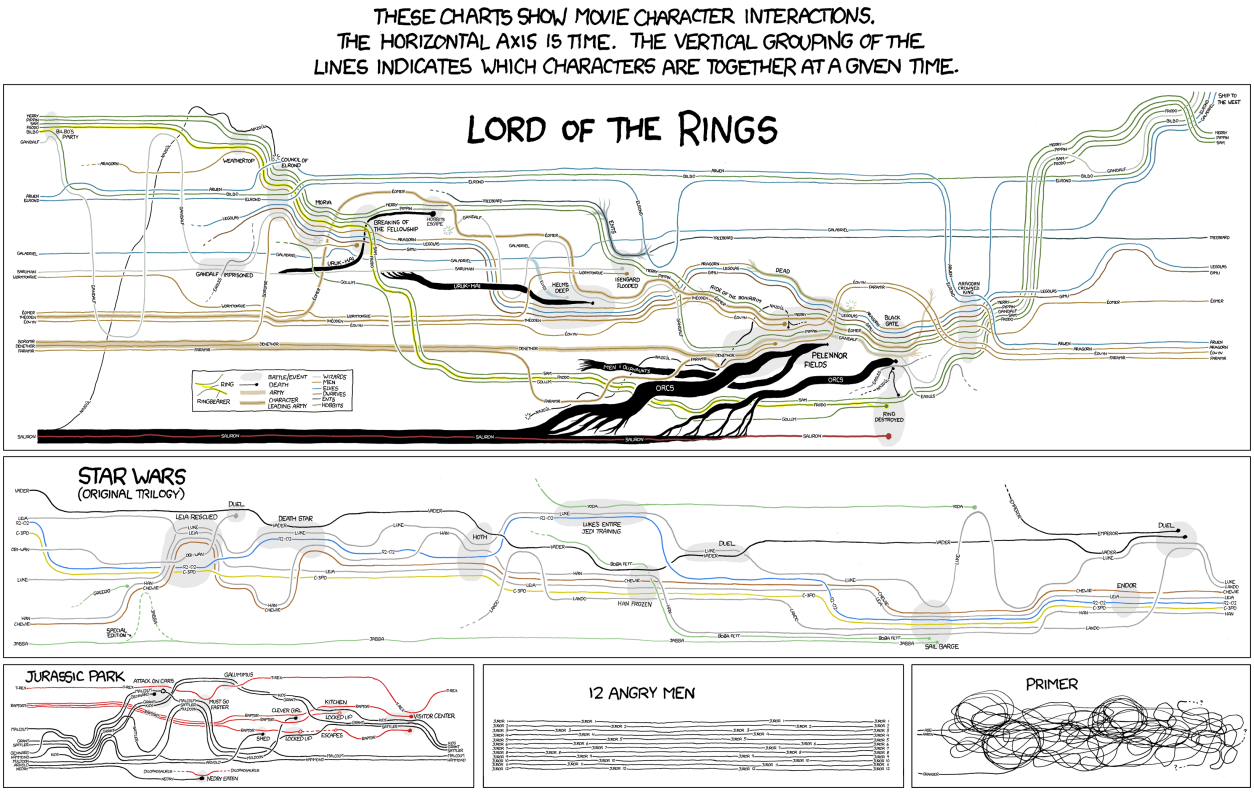


Figure 2.1: Movie narrative charts from XKCD #657. Each line corresponds to a single character. Lines that parallel one another closely represent characters moving and acting as a group.

2.2 Visualizing Interaction

The closest analogue to the kind of narrative structure we want to display is the UML Sequence diagram [28]. We show an example in Figure 2.2. The Sequence diagram shows a series of messages exchanged between entities, presumably computer programs communicating over one or more channels. We can consider this as the next layer down in our detail hierarchy. In the context of the XKCD narrative charts, sequence diagrams would appear in the gray ovals marking major events to illustrate the actual interactions that take place. These interactions might not be messages per se. One one hand, all interactions the Council of Elrond scene from the Lord of the Rings movies (save one) are spoken. On the other hand, the battles in the Mines of Moria contain a great deal of consequential interaction conducted mostly with swords and arrows. Both are perfectly valid forms of interaction.

For comparison, the UML Interaction Overview diagram (see Figure 2.3 shows a higher-level pattern of activity combining elements of flowcharts and sequence diagrams. It is similar to our proposed representation in that it represents multiple layers of abstraction at once.

2.3 Testbed Instrumentation

When one is trying to follow the detailed behavior of a program without recourse to a debugger or access to source code, one of the most informative places to start is an examination of the system calls made by the application. Since a program running under a modern multitasking operating system gains access to hardware resources by asking the kernel, these system calls and their arguments provide detailed information about what the program is doing, when, and with whom. While this technique is independent of the application being examined, its implementation is necessarily specific to different operating systems since it must talk directly to the kernel. On Linux the `strace` utility provides this information. The equivalent under Apple's Macintosh OSX is `dtrace`. FreeBSD, Solaris and UnixWare have an equivalent called `strace`. Similar capabilities exist at a higher level for software libraries such as OpenGL [24] and MPI [9]. In each case the principle is the same: by observing a program's interactions with the rest of the operating system and the outside world, much can be discerned about its inner workings even without direct access to its source code.

In addition to monitoring the actions a program takes, it is instructive to monitor the data that it sends and receives. Disk reads and writes, network sends and receives, even memory load and store operations are fair game for inspection. Of these, network sends and receives are the easiest to observe: in fact, most (if not all) current network hardware explicitly supports "promiscuous mode" where a program can ask to receive every packet that passes by regardless of its intended destination. Tools such as Wireshark [2] provide basic capabilities for capturing, filtering and analyzing TCP/IP network traffic.

The existence of these two abilities prompts research questions: given a timestamped record of network traffic and of system calls, can we attribute network traffic to the process that sent it? Can

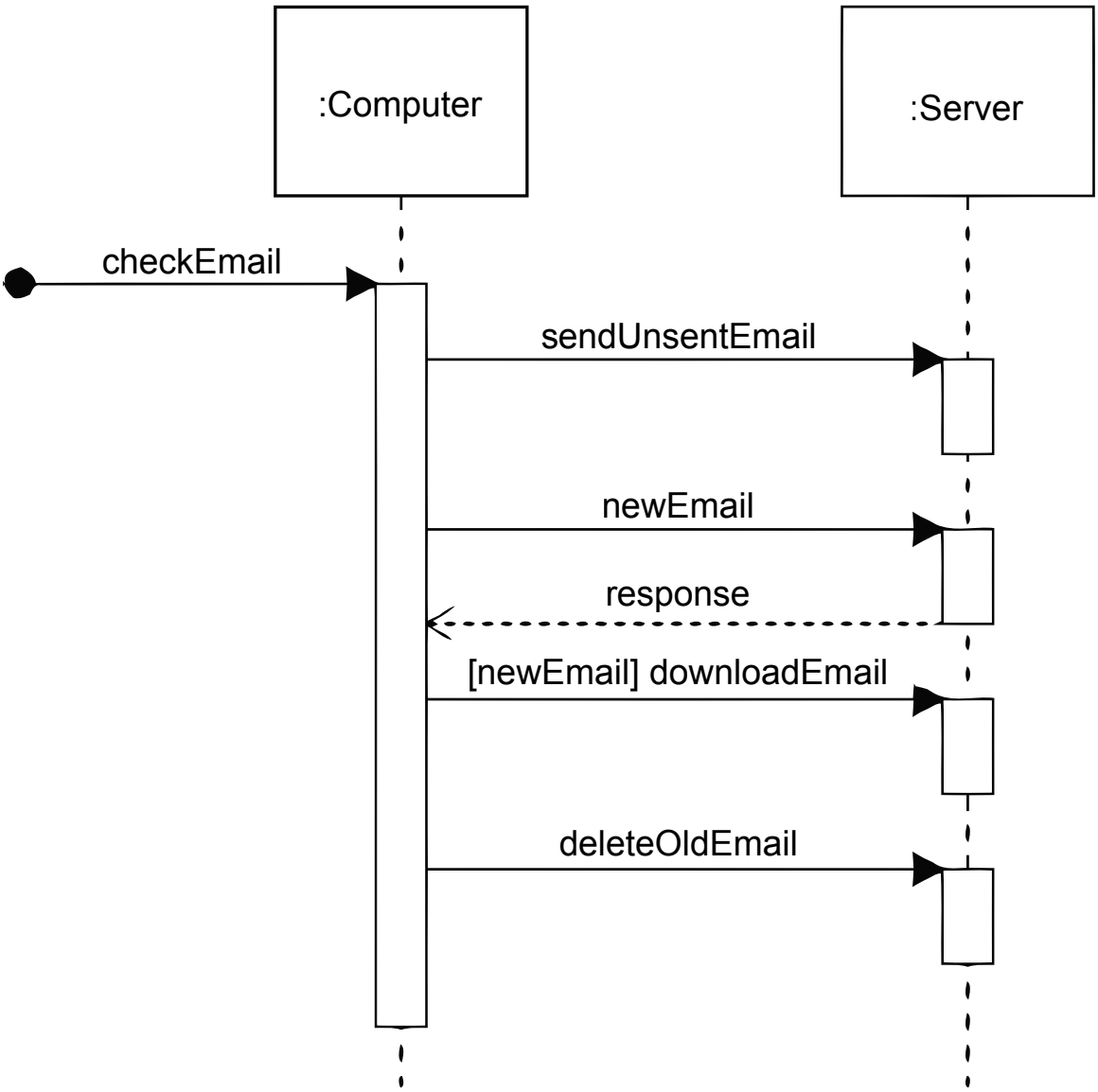


Figure 2.2: An example UML Sequence diagram showing an exchange of messages between two computers.

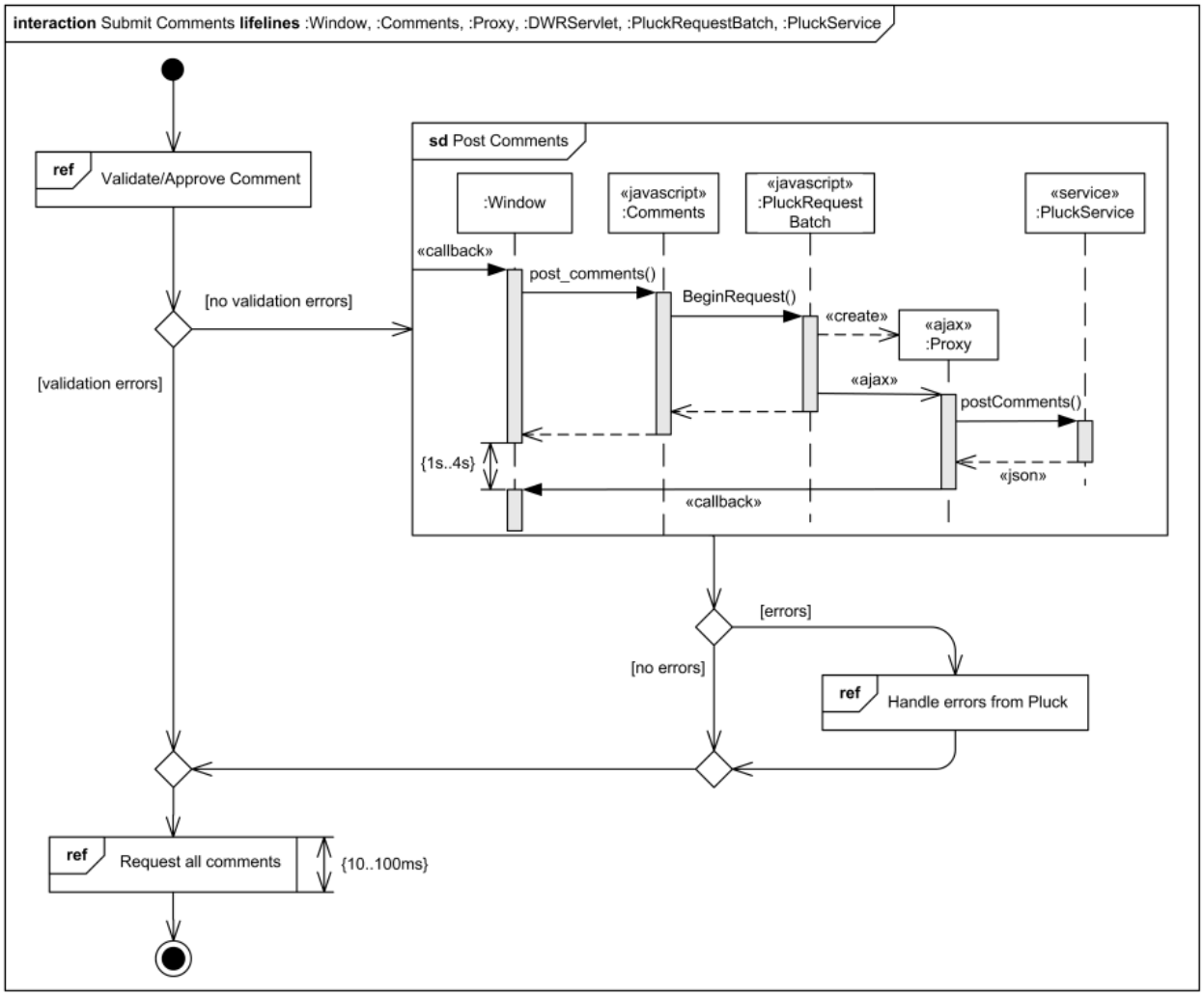


Figure 2.3: An example UML Interaction Overview diagram showing the steps a notional program goes through while submitting comments via Javascript.

we construct a causal relationship between program activity and the receipt of network traffic or actions by another program? We address some aspects of these questions in Chapter 4.

Chapter 3

Original Plans

In this section we discuss our original plans for Nested Narratives, provide an overview of the research we eventually performed, and explain the reasons for the difference.

3.1 High-Level Plans

We begin with our original aim of rendering visible the stories of cybersecurity incidents. These stories fit roughly into the following structure.

3.1.1 Narrative Levels of Detail

From lowest to highest, we envision the following levels of abstraction in cybersecurity narratives:

Data on the Wire: The lowest possible level involves individual bits and bytes in memory or in packets. These could be attack code, a payload, any data of interest to an event. Actions at this level alter the state of individual processes running on a single computer.

System-Level Actions: Narrative events at this level of abstraction are commands executed on a system. The actors are processes, themselves commanded by other processes or by humans. Actions at this level alter the state of one or more systems to transfer data, allow access or prohibit it.

Attack and Response: At this level we find humans, singly or in groups, acting to attack, defend or secure a system. These attackers and defenders initiate the actions that are visible at more detailed levels. Actions at this level are meant to retrieve or destroy information, to enhance or remove capability.

Strategic Campaigns: At this level, actors are large organizations, up to and including nation-states and their major components. Actions target the strategic interests of other actors on the grand stage. Individual commands issued on individual computers are as invisible at this scale as the flow of individual electrons when turning on a lamp.

3.1.2 Nested Narratives

We envisioned a nested representation where lower-level stories could be collapsed to show higher-level structure and expanded to show lower-level detail. Ideally, the lowest-level nodes in this graph structure would contain links to raw data. Such a narrative could itself be incorporated and encapsulated in yet another story that either linked to the events depicted or even used them whole as a smaller component in a larger setting.

3.1.3 Instrumenting Testbeds for Attribution

One of the elements of our original proposal was to automatically attribute activity on a target computer to processes (and perhaps even individual keystrokes) on an attacking system. We used Pattengale's LAASER-ptr research project as a basis for this part of the project.

3.1.4 Focus: Training for Cybersecurity

We chose to focus on tools and techniques that would help train newly hired cybersecurity team members in the art of incident response. Discussions with current staff convinced us that "art" is indeed the correct term – "a skill at doing a specified thing, typically acquired through practice" – as opposed to the systematically organized body of information that defines a "science".

Cybersecurity incident response is typically learned during a period of apprenticeship and supervised investigation. While broad and deep technical skills are essential, the true art is the ability to sniff out anomalies and clues, then assemble those into patterns and stories that suggest further action. Our hypothesis was that tools providing support for narrative construction would help formalize the training process as well as help responders in training learn more quickly.

3.2 Working With Real Data

We were fortunate to have the opportunity to work closely with Sandia's Cybersecurity Incident Response staff. From the beginning, our plan had been to embed one of our researchers within the department to watch, listen and learn. We reasoned that the most effective, applicable tools would be informed by learning the art of incident response as practitioners. Moreover, our experience on other projects has convinced us that there is truly no substitute for working with real-world data with all its complications and nuances.

One of the chief concerns with real-world data is that it carries real-world characteristics: sensitivity, confidentiality, and legal requirements for handling and disclosure. This is especially true in cybersecurity where every bit of data an organization possesses may be in play. We invite the

reader to consider all the havoc that could ensue if the right (or wrong) data set were disclosed to some external party.

While we were permitted to embed one of our researchers with the incident response staff, we were not granted access to the unfiltered source data we had hoped to use. Although the data we had access to was an extraordinarily valuable resource, it did not help us learn what we truly hoped to learn, namely the process by which an analyst distinguishes incidents worthy of investigation from false alarms or attacks that were foiled by defenses already in place. This was a fatal blow to our plans to develop tools for analyst training. We were disappointed, of course, but we accept that security needs must take precedence over research goals.

3.3 What Now?

We considered several possible approaches that might support our original goals including the following:

- **Track and render cyber alerts over their lifecycle.** We wanted to gain insight into the patterns of analysis that were most common for (1) inconsequential alerts that could be quickly marked as irrelevant, (2) events that required investigation to determine their impact, and (3) high-priority events that must be escalated and reported.
- **General narrative construction tool.** We implemented a very early prototype of a narrative construction tool with particular support for nesting. We set this aside (reluctantly) when we realized that we were spending most of our time re-implementing commercially available diagramming tools and almost no time on cybersecurity or narrative construction.
- **Make up our own tasks and data.** This approach would sidestep the sensitivity concerns surrounding real data at the risk of introducing our own biases. We chose this approach and used it to test broader hypotheses about narrative formation in the context of forensics tasks. We discuss results in Chapter 5.

Chapter 4

Instrumenting Testbeds

4.1 Abstract

Despite ever increasing adoption of distributed systems, there continues to be a dearth of general purpose tools for capturing, analyzing, displaying, and communicating the operational manifestation of complex distributed systems for other than performance purposes. Such tools could be highly useful in (for example) reducing the learning curve for new users of a distributed deployment, enhancing/aiding knowledge transfer between users or administrators of such a deployment, comparing differences between versions of a distributed software package, and comparing disparate competing packages. We present "LAASER-ttag," a prototype for noninvasively capturing the operation of distributed systems in a testbed setting. By leveraging and extending a modern Linux tracing toolkit (LTTng), we effortlessly collect and incorporate into our analyses data from different subsystems such as disk and network. Our prototype imposes no source code modification (or recompilation), and is completely agnostic to the application under study. After presenting the design and rationale behind LAASER-ttag, we show select samples of its output across a number of use cases.

4.2 Introduction

The widespread adoption of cloud computing technologies in industry means that many software users now rely on complex, distributed systems to solve their day-to-day problems, whether they know it or not. There are many instances where cloud computing technologies have made it easy for general users to take advantage of distributed systems without having to face the steep learning curve associated with traditional parallel processing architectures. Large-data frameworks such as Hadoop[25] make it easy for users to store and analyze massive amounts of data in a cluster without having to worry about the specifics of how data and computations flow through the system. Open source cluster file systems such as Ceph or GlusterFS make it easy to present a cluster's distributed storage as a single mount point that legacy web servers can utilize for scale-out storage. Infrastructure-as-a-Service (IaaS) cloud software such as OpenStack[21] provide a convenient means of provisioning a cluster's resources out to end users in the form of virtual machines. All of these technologies utilize software frameworks to manage distributed resources and simplify the

amount of work end users must do to take advantage of a cluster.

While it is important to make distributed systems more usable, quite often developers want or need to know what exactly their framework is doing under the hood. For example, performance-oriented users need to understand how resources and tasks are scheduled in a framework when refactoring applications to maximize performance. In situations where sensitive data is involved, security researchers need to be able to inspect a framework's behavior to verify that sufficient safeguards are in place and that the frameworks do not provide new opportunities for attackers. Users with high reliability requirements often need to verify that data and computations are in fact distributed by a framework in a way that the system could survive a known number of failures. Finally, application developers often want to inspect a framework's behavior to help discover race conditions and bugs in their own applications.

While there are many tools available today for analyzing different aspects of complex distributed software systems, we have yet to find one that covers all of our needs in a generic manner. The vast majority of distributed analysis tools focus on providing performance information. Ganglia, Nagios, Supermon, OVIS, and Bright Cluster Manager provide an effective means for collecting runtime performance information about applications in a cluster. Unfortunately, these statistics generally do not reveal enough information to infer a detailed understanding of a distributed application's low-level behavior. There are a variety of application-specific instrumentation and monitoring efforts for specific frameworks, including Hadoop's Chukwa and Cassandra's JMX interface, as well as approaches that simply parse a specific framework's log files. These approaches are extremely insightful for understanding applications that utilize the intended framework. However, each has its own learning overhead, and the application-specific nature of these approaches prohibits generality.

Our research is in developing tools and techniques to help rapidly understand how different distributed software frameworks behave. We argue that this work is best accomplished by finding a middle ground between capturing high-level system statistics and application-specific instrumentation: instead, use kernel-level instrumentation to generically capture important, system-level events in the life of the framework that can be analyzed offline to extract meaningful behavior over time.

We have prototyped a solution that largely achieves our goals. By leveraging and extending a modern trace framework – The Linux Tracing Toolkit, next generation (LTTng)[1], we have been able to rapidly assemble a relatively non-invasive high-fidelity platform spanning subsystems such as disk and network. We have used this platform for collecting, analyzing, and displaying the operations and interactions carried out by a variety of distributed software packages.

The basic reasoning behind leveraging a system-level trace framework is that system-level calls (e.g. syscalls) are the well-defined crossings between computer programs and various subsystems of interest, such as disk and network. Thus tracing these points is a natural and parsimonious approach for observing how applications in general treat data.

The version of the prototype covered here focuses almost exclusively on 'tag tracking,' which refers specifically to placing short prespecified strings (the so-called 'tracked tags') into input data

and subsequently observing them they traverse a cluster during distributed computations. This tag tracking prototype is part of a larger program (beyond the scope of this publication) toward 'Live All-encompassing Automated Scoring and Event Reconstruction' (LAASER) in computer network testbeds, and thus we refer to the prototype detailed here as LAASER-ttag.

The remainder of this paper is structured as follows: Section 4.3 explains general approach as well as our prototype platform in great depth. Section 4.4 shows our system in action by showcasing and discussing a variety of tag-tracking analyses. Section 4.5 comments on the implications of our tool as well as future directions.

4.3 Approach and Methods

4.3.1 High Level Design Rationale

At the highest level, our goal for this work is loosely stated. We desire a solution for recording the detailed operation of testbed cloud systems in a form which lends itself straightforwardly to high level human understanding. The solution space for this loosely stated problem is immense. The most natural starting points, perhaps, lie in using already resident system functionality such as (on Linux, at least) `netstat`, `ps`, and the `/proc` filesystem to cobble together snapshots of testbed nodes as the cluster operates. Other natural (but typically not system resident) data sources are network packet capture (e.g. `libpcap`) and filesystem watches (e.g. `inotify`). Pushing these various and disparate datasources through log aggregators such as Splunk has in fact shown promise, but in practice causes heavy system loads due to their polling nature[30].

Avoiding such performance hits is one of the reasons we chose the path leading to LAASER-ttag, which is based upon system level *tracing*. LTTng (the tracing framework we extended) works by leveraging kernel *tracepoints* – prespecified locations in kernel code which call out to functions provided by custom (LTTng provided) loadable kernel modules for dumping structured, packed binary, trace entries to disk. Much more information on system level tracing, including the list of tracepoints used by LTTng, is available via LTTng's documentation[17] or a variety of other sources[14]. Our prototype only uses a subset of these tracepoints (mainly file system and network operations).

Tracing, by design, is inherently event driven; upon events of interest control is transferred to code LTTng provides for inspecting and recording system state at that instant. Other event-driven approaches include instrumented library code for subsystems of interest and custom instrumented application code[16]. As mentioned in the introduction, we desire a 'more uniform and less invasive' solution. Now we are prepared to define these terms more precisely – by uniform we mean that it should be possible to instrument and analyze a wide variety of tools according to a common methodology and technology substrate, and by less invasive we mean that we want to avoid having to modify or recompile the source code of the tools under observation. We have achieved these goals with our prototype. As our system collects data at the operating system level, it is agnostic to the application under study. For the exact same reason, it imposes no source code modification

(or recompilations) requirements on the application under study.

The space of possible analyses enabled by trace data is large. For example, it is straightforward to produce a listing of all Hadoop (see Section 4.4.1 for a more detailed discussion of Hadoop) components annotated with process ID and role, (e.g. DataNode) along with a record of all of the files that they accessed during a distributed computation. Unfortunately, even for simple computations, this listing can be large and difficult to visualize. There are many strategies worth exploring for managing the complexity (and sheer size) in such general purpose analyses. However, we chose a different path, and focused our attention on a rather simple analysis – putting tracked tag observations into the context of operations that were handling them. This analysis in our experience has a wonderful filtering effect, and renders our datasets manageable in size. That we can present both meaningful and readable timeline graphics (e.g. Figure 4.2 in Section 4.4.1) on normal sized paper is evidence of this filtering effect.

As we subsequently learned throughout development and testing of LAASER-ttag, even tag tracking presents many challenges. Foremost is comprehensiveness, for example, in order to read from disk, applications have a choice in the routine they employ. They can use the well-known `read` call, they can use the scatter-gather `readv`, or they can use `mmap` to map the file and read it as if it were in main memory, among others. In order for LAASER-ttag to comprehensively catch *every* traversal of tracked tags through subsystems of interest requires covering all of the independent paths that data can take through a system. Given the limited scope and funding for LAASER-ttag development, we chose essentially to defer to LTTng in selecting a sufficient set of tracepoints, and deal with blind spots as they arise. For example, Section 4.4.2 details a known blind spot of LTTng, memory mapped file I/O.

4.3.2 Trace Gathering and Pipeline Specifics

To conduct the analyses described in this study, we use a (locally) modified version of the Linux Tracing Toolkit next generation (LTTng)[1]. We made modifications to the 0.19.11 LTTng loadable kernel modules to enable 'tag tracking.' More specifically, we have enhanced the LTTng modules to search for each member of a predetermined fixed-size array of strings (specific strings shown for reference in Table 4.2) upon calls to, e.g., `fs.write`, `fs.read`, `net.socket_sendmsg`, `net.socket_recvmsg`. By strategically seeding input data with instances of strings from the predetermined list, our LTTng modules enable reconstructing high fidelity synchronized[23] timelines of data traversal through network cards and file systems of an instrumented cluster during distributed computations.

In Table 4.1 we outline the versions of the various LTTng components used in our current prototyping cluster. For simplicity, our traced clusters (so far) have mainly been homogeneous populations of Ubuntu 11.04 virtual machines. The version of LTTng that was available when we were building our prototype (LTTng 0.19.x) required kernel patches that are no longer required by newer generation LTTng (2.x) releases. As such we patched Ubuntu's 2.6.38-9 build with LTTng's 0.249 kernel patch set (written against mainline kernel 2.6.38.6).

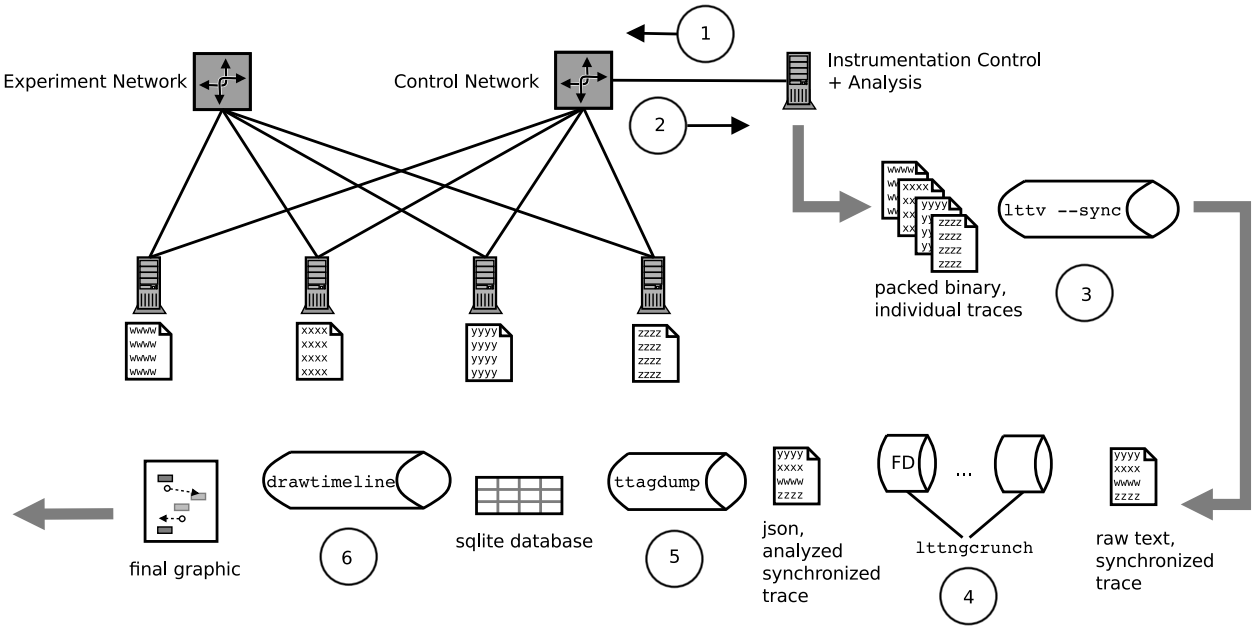


Figure 4.1: A sketch of our overall collection scheme and data processing pipeline

The machine suite used for the analyses in this paper consists of a collection of virtual machines. All but one of the machines (the cluster) perform the distributed computations, and other than normal system software contain installs of the various distributed software packages as well as LTTng. The additional node is the *instrumentation control and analysis* machine, and contains scripts for batch-controlling tracing on the cluster as well as analyzing the collected traces.

Figure 4.1 depicts our collection scheme and analysis pipeline, and works as follows:

1. After receiving a command from the *instrumentation control* node, individual cluster nodes begin tracing and storing results to local disk. In our current prototype, the commands are issued via ssh with a command such as `ssh nodeX lttctl -C sampletrace -w /home/ltt/sampletrac` We prefer that the control box reside on a separate control network, such that commands arrive at (and trace data leaves) cluster nodes via an independent network interface than cluster inter-node network traffic proper. This affords stronger experimental pedigree as the two types of network traffic are not commingled.

2. Once an experiment has concluded (or periodically, for long running experiments), the control box commands each cluster node to stop tracing and proceeds to download the trace results, in their packed binary form, for subsequent synchronization and analysis. Our current prototype simply uses scp for downloading individual traces. In the future we intend to explore LTTng's streaming capability.

3. Individual node traces are globally synchronized by LTTng's lttv (trace viewer) tool. This ability to globally synchronize traces is another notable attribute of, and one of of the major reasons that we chose, LTTng. Their global synchronization method is detailed in [23], and

essentially amounts to a clever implementation of [7] using each cluster node's TSC (timestamp counter) register as a local timestamp along with inter-node events having a known ordering (TCP packet transmit/receive) to find a globally consistent linear mapping of each node's TSC values to a global time. For simplicity, we store the globally synchronized event transcript in lttv's textDump format. This text-based format is space inefficient (especially relative to LTTng's binary format), but has been manageable so far. An example line from the globally synchronized trace is as follows:

```
fs.read:  356245.554562472 (/home/.../hdfs1/fs_0), 18788,
18766, /.../bin/java, , 18766, 0x0, SYSCALL { count = 545, fd = 5, therep
= 4194312, ret = 545 }
```

This event indicates that a filesystem `read` was carried by java (pid 18788, tgid 18766, parentpid 18766) on node 'hdfs1' resulting in a buffer full of 545 characters which were pulled from tgid 18766's fifth file descriptor. The 'therep' field is detailed below.

4. As can be inferred from the example trace line above, accumulating state is necessary to put any individual event into context. For example, to appropriately ascribe the example `fs.read` to a meaningful filename (or socket, or pipe) requires keeping track of file descriptor creation events. The corresponding event in this case is as follows:

   ```
   fs.open:  356244.909153060 (/home/.../hdfs1/fs_0), 18788,
   18766, /.../bin/java, , 18766, 0x0, SYSCALL { fd = 5,
   filename = "/home/ltt/gettysburg.txt"}
   ```

   To accumulate this state information across the various cluster machines, we have written a highly modular tool called `LTTngcrunch`. For tag tracking, LTTngcrunch's operation is fairly simplistic. It consumes the globally synchronized textual output as shown above, parses it into an object representation, which is then passed through a pipeline of user-specified modules. For tag tracking, our events pass through modules which perform file descriptor bookkeeping (for regular files and TCP/IP sockets) and process bookkeeping. Bookkeeping refers rather simply to accumulating (python) dictionaries of system state (e.g. file descriptor tables), in order to decorate an event's object representation with more comprehensive information (such as a filename instead of merely a file descriptor number). Thus, in later stages of the pipeline, the data need not be traversed serially in order to retrieve corresponding state. The output of LTTngcrunch, for ease in portability, is in javascript object notation (JSON). With some fields omitted for brevity and readability, the following is an example of an LTTngcrunch output object:

   ```
   {count:545, event_type:fs.read, tgid:18788, pid:18766
   local_timestamp:356245.554562472, fdext:"/home/ltt/gettysburg.txt", ret:545,
   seqid:431501, fd:5, therep:4194312}
   ```

   For tag tracking, the most important field in these objects is the 'therep' field, which reveals whether tracked tags were seen in this event. The title 'therep' is meant to be interpreted as a predicate (as in predicate logic), i.e. 'is it there?' In this case 'it' refers to tracked tags, and therep is interpreted as a bitmask. In other words, therep being nonzero implies that a tracked tag was seen in the corresponding operation. For example, 'therep=4194308' in an `fs.read` event means that the tracked tags 'ulr821' and 'fix283' were seen in the buffer

component name	version	description
lttv	0.12.37-17022011	Visualizer
ltt-control	0.88-09242010	Trace daemon, etc.
ltt-modules	0.19.11 (+ in house mods)	Kernel modules
patches	0.249 (targeting 2.6.38.6)	Kernel patch set

Table 4.1: LTTng tool versions used in our prototype

being returned by `fs.read` since $4194308_{10} = 00000000100000000000000000000100_2 = 2^{22}_{10} + 2^{2}_{10}$ and 'fix283' is the 22^{nd} and 'ulr821' the 2^{nd} zero-indexed entries, respectively, in our prespecified tag array (Table 4.2).

5. The final data refinement step in our pipeline is to store all of the (now JSON formatted) events where therep is nonzero in a SQLite database (with a schema detailed in [5]). This is accomplished by a fairly simple python script which consumes JSON objects, and writes data to a SQLite database, appropriately formatted per our schema. At this point we consider our analysis complete, and the SQLite product is amenable to interpretation in any way desired (perhaps, most easily, as a spreadsheet). It is of note that the number of events where therep is nonzero is typically orders of magnitude smaller than the original number of traced events, and as such the resulting SQLite files are typically small.

6. The standard fashion in which we inspect the SQLite files produced by our pipeline is via an in-house developed timeline generator. We will see a number of examples of these timelines in subsequent sections (e.g. Figure 4.2 in Section 4.4.1), which depict (global) time on their vertical axis and contain a column for each process (thread group id (TGID), more specifically) that handled a tracked tag. This medium has proved natural for understanding node-to-node interactions, as well as intra-node operations, where tracked tags are involved.

4.3.3 Limitations

Our system also suffers from a number of minor limitations:

- The 0.19.X LTTng kernel patches place tracepoints at the locations where traced functions are about to return control to their callers. In general this is not a problem, but in certain cases makes for difficulty in deciphering results. For example, if a socket is sending data asynchronously (i.e. with the socket option O_NONBLOCK set), the net.socket_send event typically occurs before the corresponding net.socket_receive at the other end of the socket. This is contrary to the order a user of LTTng comes to expect, because normally the net.socket_receive will *return* before the net.socket_send (which blocks until receipt is confirmed).

- While the global time synchronization feature of LTTng is certainly distinguishing, it is not without its own limitations. For example, every node must exchange traffic with every other

node at least once during each tracing session (albeit only a few packets for each node pair). This all-to-all communication requirement scales quadratically with number of nodes, and may be prohibitive in large clusters.

- Tracing has the potential, especially on highly utilized systems, to produce huge amounts of data. We save approximately one order of magnitude in storage requirements (versus LTTng in its standard configuration) by selectively deactivating tracepoints which are non-essential to our analyses. This savings has been sufficient to enable all of the experiments we have conducted to date. If larger savings are needed in the future, it will not be prohibitively difficult to modify LTTng to selectively save events produced by, e.g., white listed applications. This is only one of many potential space saving strategies.

4.3.4 Scanning Algorithm

A key challenge in developing an effective tagging systems is implementing an efficient system for inspecting data that moves through the instrumentation points. Our needs require that a small number (30) of fixed-length (6) strings be used as a search dictionary, and that tags can start at any position in the stream. Since we have control over the tags used in a system, we can simplify the search task by using non-overlapping tags that remove the need for tracking multiple potential hits at the same time.

We considered multiple strategies for string matching in the streams. While efficient algorithms such as Boyer-Moore and Knuth-Morris-Pratt would be ideal, we constructed the simple but effective approach listed in Algorithm 1. This scanning algorithm was straightforward to implement and met our performance objectives. It is invoked in the following LTTng tracepoints to check for the existence of tracked tags in input/output buffers:

Algorithm 1 Simplistic scan for fixed length prespecified strings in a buffer. While this routine has $O(nmw)$, we reasonably assume m and w as constant. Further, n is also typically small, and thus this strategy is not time-prohibitive.

Require: a character buffer of length n
Require: a tag array of length m, containing tags with fixed width w, e.g. Table 4.2
Ensure: a bitmask b where bit i being set indicates that tag i exists in the buffer

```
1:  function SCAN-FOR-TTAGS(buffer)
2:      for all positions i from 0 to n − w + 1 do                ▷ O(n)
3:          for all tags t with pos j in tag array do             ▷ O(m)
4:              if t =buffer[i : i + w] then                      ▷ O(w)
5:                  b ← b ∨ 2^j                                   ▷ ∨ denotes bitwise OR
6:              end if
7:          end for
8:      end for
9:  return b
10: end function
```

0 yqz958	1 wbu365	2 ulr821	3 jrs036	4 rkf168	5 jxm820
6 ori894	7 yko871	8 ftu070	9 srf502	10 grl148	11 lyr428
12 dpp223	13 roc357	14 ddj250	15 vio154	16 pzz933	17 bjk412
18 wqv139	19 yvl354	20 wfb150	21 bwj563	22 fix283	23 ogd030
24 oie495	25 ggh069	26 wyc894	27 hpn120	28 riu782	29 bbt515

Table 4.2: Current prototype's tracked tags array. For example, if *fix283* is found in a buffer, Algorithm 1 will return a bitmask with the 22*nd* 0-indexed bit set. In the notation of Algorithm 1, $m = 30$ and $w = 6$.

4.4 Case Studies

We now present a variety of LAASER-ttag analyses in order to illustrate its utility in understanding cluster operations. The scenarios covered in this paper are intentionally short, simplistic, and involve only a few processes across a small number of cluster nodes. That they are short is in an effort to save space, but not at the expense of showcasing a meaningful set of operations.

4.4.1 Hadoop

The case studies in this section were conducted with Apache Hadoop (http://hadoop.apache.org/). Hadoop is a framework for distributed processing of large data sets. Hadoop has two primary components: MapReduce, which is patterned after Google's MapReduce, and the Hadoop Distributed File System (HDFS)[25] which is patterned after Google's GFS. HDFS is a replicated block store and functions as the storage layer of the Hadoop framework. Hadoop MapReduce runs a TaskTracker process on each node for processing data. Its JobTracker process manages the processing tasks. HDFS's NameNode server process runs on a single node and stores the metadata (file names, permissions, replication factors, etc.) for all the files in the file system. The SecondaryNameNode process assists the NameNode in compacting the metadata stored on disk. File content is divided up into blocks and stored by the DataNode processes running on all the nodes in the cluster.

The case studies in this section were conducted in a cluster where DataNode and TaskTracker processes were running on every node and the NameNode, JobTracker, and SecondaryNameNode processes were running on one of those nodes. The HDFS replication factor was set to two, which means that each HDFS block will be replicated to at least two distinct nodes.

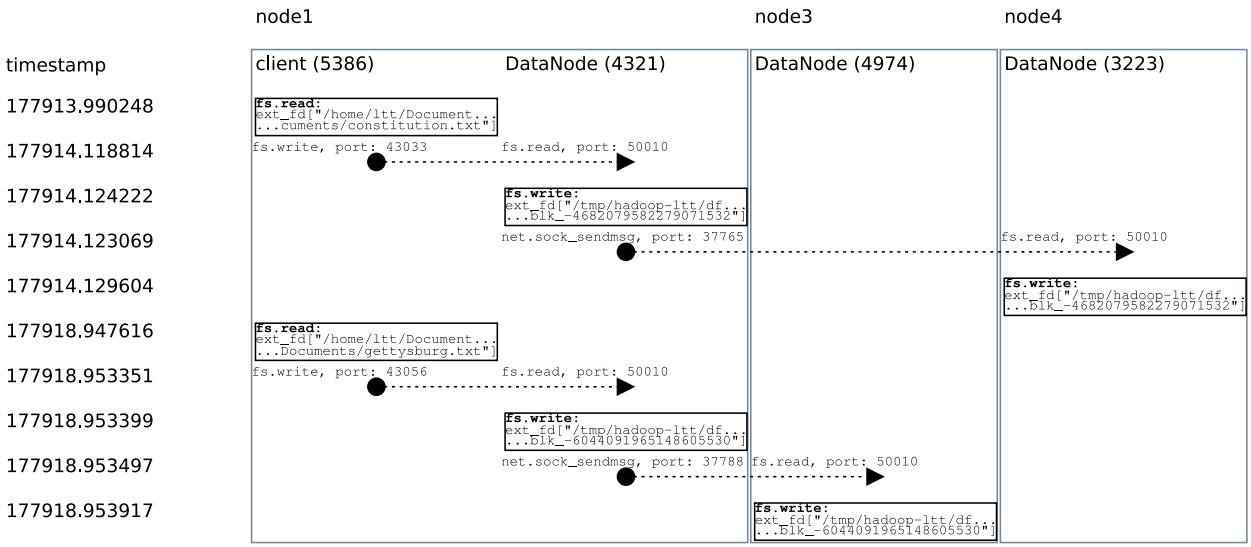


Figure 4.2: A Hadoop client puts two files into an HDFS by submitting them as blocks to its local DataNode. In turn, the local DataNode replicates each block to a second DataNode somewhere else in the cluster.

Simple Tag in Data File

In our first case study, we traced movement of file content in HDFS. HDFS writes are performed by a client process that reads files from the local disk and then sends the file metadata to the NameNode. The client's communication with the NameNode is over Java Remote Procedural Calls (RPCs). The NameNode returns to the client a list of DataNodes to write each file block. The client then sends each block and its DataNode list to the first DataNode in the list using a custom binary protocol. The DataNodes replicate the blocks they receive to the next DataNode in the provided list.

Figure 4.2 shows the output of LAASER-ttag after executing

```
bin/hadoop dfs -copyFromLocal
   ∼/Documents/
   /user/ltt/gutenberg
```

from one of our cluster nodes. This command copies all of the files used in this Hadoop tutorial[19], as well as two additional files seeded with tracked tags (the Gettysburg Address and the U.S. Constitution) into an HDFS directory with path `/user/ltt/gutenberg`. The picture shows that the cluster is configured to replicate blocks to two nodes, and that the replication is done as a relay (as opposed to a broadcast). Second, it gives hints as to HDFS' block structure and naming scheme, as the block file names and underlying directory structure imply some sort of hashing scheme.

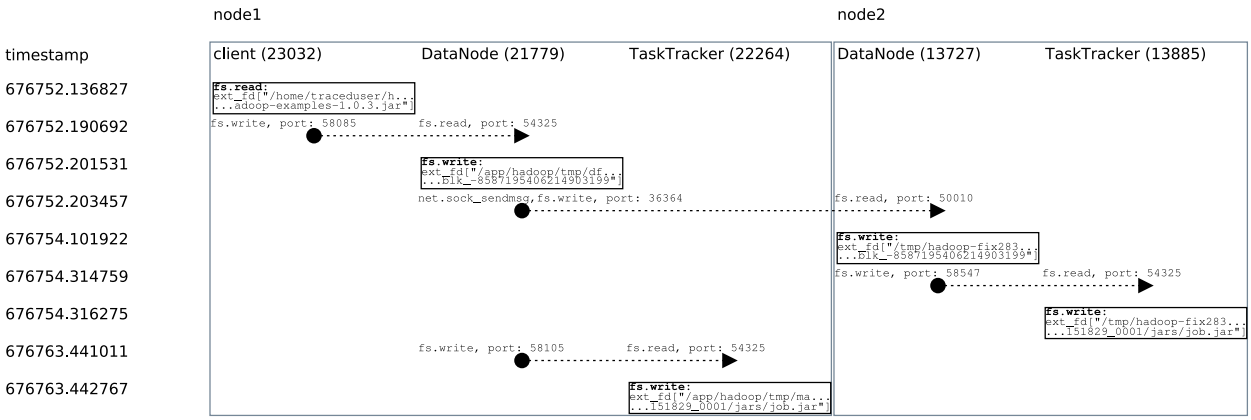


Figure 4.3: Upon invocation of a MapReduce job (by the client on node1), Hadoop distributes copies of the tagged Jar file to TaskTrackers across the cluster.

Tag as Username

By placing tracked tags in locations other than input data, we are able to learn about other aspects of the system under study. In this case, we set the username of our HDFS transaction to be a tracked tag. By invoking the same command as in the previous subsection (Section 4.4.1), we observe that in the default HDFS configuration, the username rarely manifests in the network or on disk. Specifically in this case, the username only traversed the network as the HDFS client initiated conversation with the NameNode, and only manifested on disk as the NameNode updated its filesystem journal (`dfs-root/name/current/edits`). The SecondaryNameNode did not compact the metadata on disk during the trace.

In contrast, the username manifests much more copiously in a fairly simple MapReduce job. When a MapReduce job is submitted by the Hadoop client, the JobTracker instructs the TaskTracker which part of the job to process. The job's code and configuration settings are distributed to the TaskTrackers via HDFS. By cursory inspection, the username manifests in a dozen TaskTracker configuration and job specification files, another dozen logs, and in blocks across all nodes of the distributed file system. We omit detailed trace graphics for this case due to space constraints.

Tag in Code

As a final case for HDFS, we seeded executable code (in the form of a Java jar file) with a tracked tag in order to examine HDFS from yet another angle. Figure 4.3 shows the analysis produced after running MapReduce job from the previously mentioned Hadoop tutorial [19]. The figure shows the distribution of the jar file from the client to the DataNodes and then from DataNodes to the TaskTrackers. This analysis, of all we have conducted to date, gives us hope that it will be possible to use LAASER-ttag data to infer highly abstracted characterizations of distributed systems at the distributed-protocol level.

4.4.2 Tag in MapReduce job input (with the details of Memory Mapped File I/O shortcomings)

As discussed earlier (Section 4.3.1), while LAASER-ttag makes a best effort at comprehensively tracking data as it traverses through our clusters, the current prototype undoubtedly has blind spots. As an instructive example, we were suspicious of one of our early analyses (of a MapReduce job, analysis figure omitted due to space constraints) in that there were network `send`/`recv` events of tracked tags without preceding filesystem reads. To wit, how can Hadoop send HDFS blocks out to the network without reading them first from disk? Digging lightly into the HDFS source revealed that the BlockSender class (`BlockSender.java` in `hdfs/server/datanode/`) reads HDFS blocks from disk via Java's "new" I/O (`java.nio.*`) FileChannel class, which is Java's abstraction for memory-mapped file I/O. Since our current prototype ignores memory-mapped file I/O (mainly because LTTng 0.x does not provide a natural tracepoint for `mmap` and friends), we essentially missed the disk reads. Rerunning the same command on an older version of Hadoop (namely 0.17.2, the last version that did not use Java FileChannel) yielded an analysis in which our system correctly observes the disk reads before the network sends.

4.4.3 GlusterFS

GlusterFS[8] is a popular open source distributed file system (DFS) that enables users to aggregate a cluster's distributed storage resources into one or more mountable volumes. The software is comprised of a daemon for servers that manages local storage resources, and a FUSE software interface for clients that enables end applications to transparently connect to the data maintained by the system. Unlike other DFSs, GlusterFS does not utilize a metadata server to handle data placement within the cluster. Instead, it computes a hash of a file's filename to determine which servers in the cluster are responsible for maintaining the desired data. GlusterFS can be configured to stripe (discouraged) and/or replicate (encouraged) data across multiple storage nodes in the system. If striping is not utilized, a file is stored in its entirety on a single storage node, as well as every other replicate storage node. Many users favor GlusterFS because data files are generally stored as-is in the underlying storage devices, and could easily be recovered if the GlusterFS system were to suffer a catastrophic failure.

Figure 4.4 shows a simple scenario where one cluster node (node2) performs a `sha1sum` of a file that resides in the Gluster file system and turns out not to reside on the local disk, necessitating a network transfer. As was the case with HDFS above, this picture can be quite informative to those not deeply familiar with Gluster's inner workings. First, it is straightforward to infer from the reads and writes to/from `/dev/fuse` that this Gluster deployment is via a FUSE filesystem. Second, Gluster appears to use a client-server model for retrieving data from remote nodes, as 'glusterfs' on node2 reaches out to a daemon 'glusterfsd' on node1 which in turns invokes its local 'glusterfs' on node1.

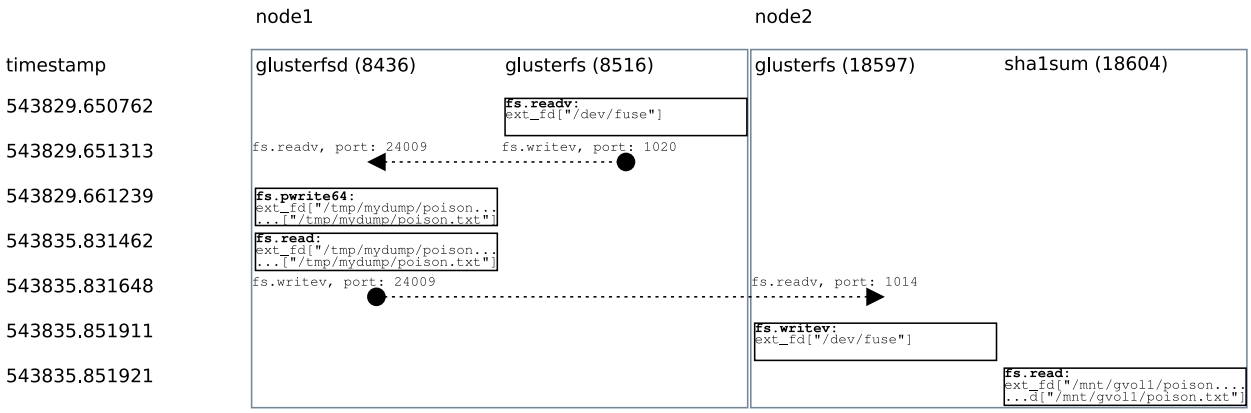


Figure 4.4: Taking the hash (sha1sum) of a file resident in a Gluster filesystem in this case triggers the underlying DFS to retrieve the file via the network.

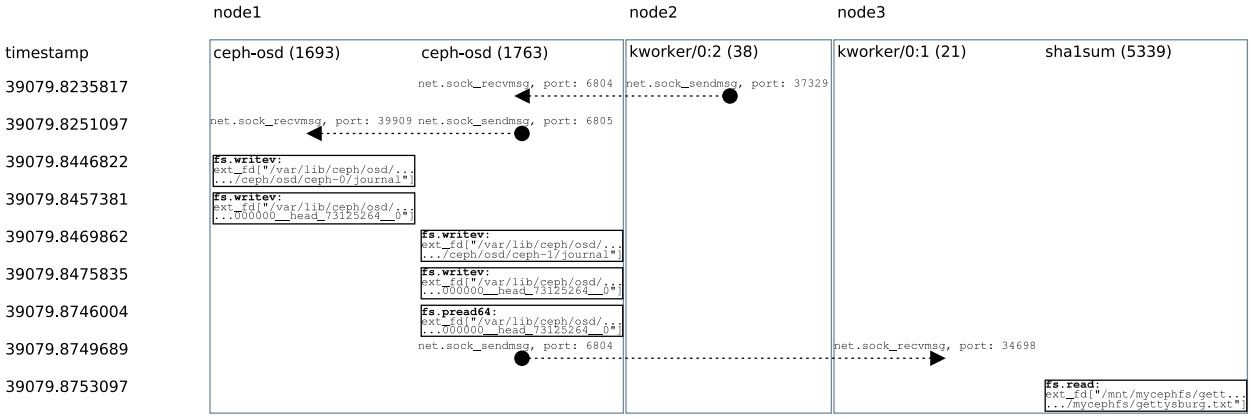


Figure 4.5: Taking the hash (sha1sum) of a file resident in a Ceph filesystem in this case triggers the underlying DFS to retrieve the file via the network.

4.4.4 Ceph

Ceph[31] is a relatively new DFS that has received a fair amount of recent attention due to the inclusion of its client interface code in the Linux kernel. Ceph was designed to be a distributed object store upon which additional storage services such as a DFS could be layered. This object store decomposes objects into smaller (8MB) blocks that are internally replicated on multiple storage nodes within the cluster.

The analysis is shown in Fig. 4.5 presents a scenario nearly identical to that shown in GlusterFS (Sect. 4.4.3 and Fig. 4.4) whereby an invocation of `sha1sum` of a file necessitates network transfer of the desired content via the underlying distributed file system. In contrast with GlusterFS, the Ceph client is part of the Linux kernel, as is evident by the kworker thread contacting the ceph-osd for the file in question. Additionally, the analysis makes evident that Ceph is journaled, and writes to the journal are done lazily (upon request, rather than upon insertion into the filesystem).

4.5 Discussion and Conclusions

Our research is in developing techniques to help rapidly understand how different distributed software frameworks behave. We have argued that this work is best accomplished by finding a middle ground between capturing high-level system statistics and application-specific instrumentation: instead, use kernel-level instrumentation to generically capture system-level events in the life of the framework that can be analyzed offline to extract meaningful behavior over time.

We have presented a prototype system, LAASER-ttag, for noninvasively and uniformly making sense of data flows throughout distributed system testbeds. As our system collects data at the operating system level, it is agnostic to the application under study, and imposes no source code modification (or recompilation) requirements on the application under study. We have also presented a number of simple case studies carried out by LAASER-ttag, thereby illustrating its utility.

In addition to diving deeper into any particular distributed software package (natural follow on steps), there are many other potential uses for LAASER-ttag.We plan to explore the space of different analyses. That is, while tag tracking is no doubt useful, we are interested in characterizing other aspects of distributed systems operational manifestation. For example, it would be worthwhile to explore how the overwhelming number of operations being carried out by a distributed system every second could be portrayed to human in a summarized but insightful fashion.

Another direction is in exploring LAASER-ttag as an independent data source for finding indicators of anomalous system activity. We envision pushing LAASER-ttag data through log aggregators along with more standard system security logs in order to statistically associate known violations (i.e. the ground truth provided by LAASER-ttag) with non-obvious but discriminatory side-effects (i.e. non-obvious indicators that will reliably manifest in fielded systems).

Chapter 5

Narrative Support for Forensics

Criminal forensic analysis involves examining a collection of clues to construct a plausible account of the events associated with a crime. Investigators typically have a relatively sparse set of clues and their task is to apply inferential reasoning to formulate alternative interpretations and deductive reasoning to arrive at a conclusion regarding the most likely account. From a cognitive perspective, several processes are involved. The investigator must interpret clues and recognize associations between clues based on general and specific domain knowledge combined with relevant past experience. Clues must be combined to form a narrative that includes basic narrative components such as the entities, their respective motives, the time and place of events, and intentions and causation [32]. Narratives must undergo critical evaluation and appraised with respect to the investigator's confidence in alternative narrative interpretations. Forensic analysis is a mentally demanding activity. With competent professionals, the prevalence of cognitive biases has been documented, with these biases present despite rigorous standards of practice [15, 4].

Research addressing cognitive factors influencing performance in forensic analysis has focused on traditional law enforcement, with somewhat less attention to medical forensic analysis. Over the past decade, there has been a gradual shift with law enforcement devoting increasing resources to cyber crime. These investigators apply similar techniques and practices, yet the clues consist of transactions occurring within digital networks, utilizing digital devices. Law enforcement is joined in this endeavor by forensic analysts working in industry and government who are often on the front lines defending information networks from criminal activities. In many organizations, cyber security positions previously filled by individuals with network administration skills have been expanded to encompass forensic analysis. Investigations undertaken following network security breaches and attacks on network resources are comparable to those undertaken by law enforcement personnel. Provided a collection of clues, the cyber security analyst must interpret events to construct a narrative account of the criminal perpetrator, and their objectives, motives, techniques and capabilities.

With the increasing prevalence and reliance on information networks, there is a growing demand for professionals capable of conducting cyber forensic analysis. However, a gap exists in the supply of qualified professionals and the demand for their services. Furthermore, for the most seasoned cyber security analyst, forensic analysis can be a difficult and demanding activity. Consequently, there is need for training and technologies that will accelerate the rate at which individuals are able to attain proficiency and enhance performance for cyber forensic analysis.

The research described in this chapter was undertaken to gain a greater understanding of the cognitive processes that underlie criminal forensic analysis, and particularly the use of narrative in the analysis cyber crimes. It is asserted that narrative construction is vital to effective forensic analysis and hypothesized that technology interventions that facilitate and promote the development of narratives will lead to superior performance.

5.1 Methods

5.1.1 Subjects

Subjects consisted of 52 employees of Sandia National Laboratories who responded to a company-wide announcement soliciting volunteers to participate in a research study concerning criminal forensic analysis. Seven subjects were eliminated due to the data files associated with their narrative analysis being corrupted and unreadable. An additional six subjects were eliminated due to their scores on the OSPAN measure of working memory [29] being 1.5 standard deviations below the mean. The characteristics measured by the OSPAN test are less important for success in those subjects' day-to-day jobs but were critical for the narrative analysis task in our study.

5.1.2 Procedure

Subjects completed the series of activities discussed in the following sections, in the corresponding order.

Narrative Experience Survey

On a scale from 0-4 (0 = "no experience" and 4 = "extensive experience"), subjects reported their level of experience for six activities involving forensic criminal investigations: (1) cyber security forensics; (2) law enforcement criminal forensics; (3) accident, root cause, event or other similar workplace investigations; (4) reading literature involving criminal investigations; (5) watching television shows, movies or other entertainment involving criminal investigations; and (6) playing computer, board or other games that involve criminal investigations.

Operation Span Task (OSPAN)

The OSPAN [29] served as a measure of the working memory capacity of subjects. In previous studies, the working memory capacity, and specifically, performance on the OSPAN, has been found to correlate with individual performance for a variety of complex cognitive tasks [3]. In the OSPAN, subjects are given a series of math operations to solve followed by a word to remember.

After each set of math operations, the subjects are required to recall the words in the order that they were presented. The OSPAN generates multiple measures of performance with the absolute and combined scores representing summary scores based on the accuracy and speed of subject responses.

5.1.3 Forensic Analysis

A scenario was composed based partially on publicized reports of actual cyber crimes. The scenario involved a fictitious pharmaceutical manufacturer and subjects were given the pretense that they had been asked to investigate a series of suspicious events at this company. Appendix A.1 provides the background information that was read to subjects prior to beginning the narrative analysis. The scenario involved three separate crimes committed by three distinct entities operating independently of one another and with different motives and objectives (See Appendix A.2). The first scenario involved a Hacktivist group intent on proving the pharmaceutical company was involved in controversial activities (i.e., biological weapons research). The second scenario had a criminal organization that committed bank fraud in which funds were redirected from used by the company. The third scenario consisted of intellectual property theft by an employee of the company (i.e., Insider). For each crime a collection of clues were identified that would be available to a forensic investigator. There were a total of 16 legitimate clues with the Hacktivist thread being the more complex having 8 clues, and the Criminal and Insider threads being somewhat simpler with 4 clues each (See Appendix A.3). There were eight additional clues that served as "red herrings" and had nothing to do with the three crimes (See Appendix A.3.2). Each laminated card presented a one sentence description of the event and the date on which the event was noted, which did not necessarily reflect the date the event occurred. Two cyber forensic analysts reviewed each scenario and verified that the storyline and clues were plausible and representative of the types of crimes a cyber forensic analyst might realistically encounter.

For the forensic analysis, subjects were randomly assigned to one of three experimental conditions (Narrative, Association and Impoverished). Elimination of subjects due to either corrupted data files or low OSPAN scores resulted in a total of 14 subjects in the Narrative, 12 in the Association and 13 in the impoverished condition.

Narrative Condition

Subjects were provided the 24 laminated cards with magnetic backings on which the clues and associated dates were printed and asked to conduct their analysis using a 57"x 46" magnetic whiteboard. Subjects could arranged the clues by affixing them to the whiteboard, and used dry erase markers (black, blue, green and red) to draw links between clues and boundaries encircling groups of clues, as well as make notes and other markings. As shown in Figure 5.1, features were provided to facilitate and encourage subjects to construct a narrative based on the clues. Primarily, the narrative features included 5 Criminal Entity Cards with labeled spaces for subjects to use dry erase markers denote the identity of the entities, "What trying to do?" and "Why trying to do it? And

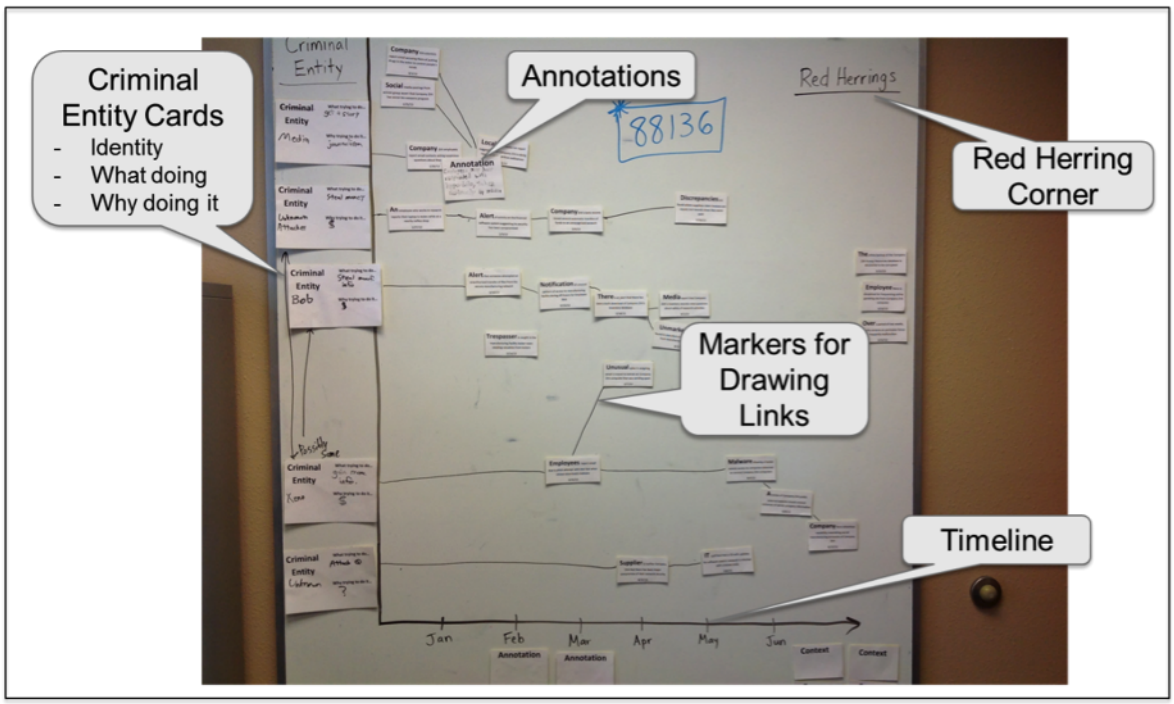


Figure 5.1: Example of the whiteboard configuration and features provided to subjects in the Narrative condition. Magnetic markers that could be used as tags are not shown here.

a timeline spanning a timeframe encompassing the dates provided with the clues. Additionally, the top right corner of the board was labeled "Red Herrings" to encourage subjects to segregate legitimate and red herring clues and subjects were given 12 annotation cards on which they make notes, 8 context cards to identify contexts, and circular magnets to use as tags with 5 different colors (white, blue, green, yellow, and red) and 6 magnets in each color (total of 30 magnets). The board was also had a vertical axis labeled, "Criminal Entities" and a horizontal axis with the timeline with months of the year denoted as tick marks.

Appendix A.4 contains the instructions that were read to each subject. After reading the instructions, the experimenter briefly demonstrated how subjects might use each feature offered in the Narrative Condition. The subject was also informed that if they had questions about the clues, they could ask the experimenter, but was not guaranteed a direct answer. Once subjects had indicated they understood the assignment, they were given a box with the clues arranged in a random order and allowed 25 minutes to conduct the analysis. Once complete, a photograph was taken of the diagram produced on the whiteboard.

Association Condition

The Association condition provided the same visuospatial elements as the Narrative condition, but without elements to facilitate and encourage construction of a narrative. The same laminated cards

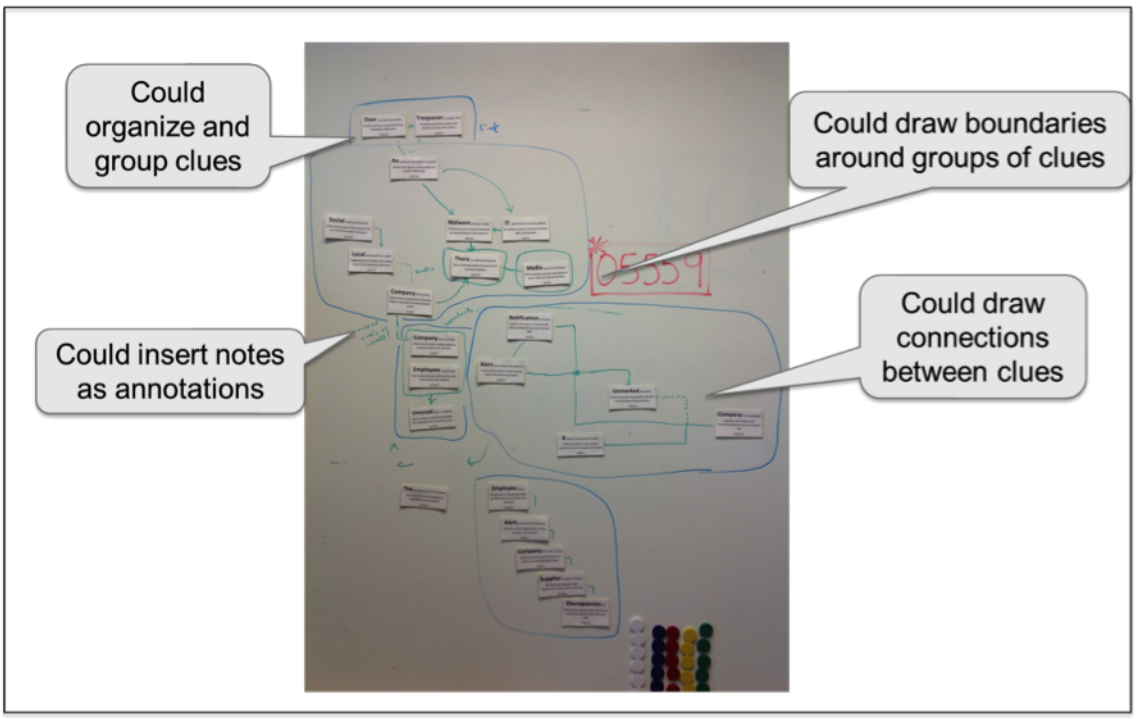


Figure 5.2: Example of diagram created in conducting forensic analysis during the Association condition.

with clues were provided and work was completed at the whiteboard. However, subjects were only provided with dry erase markers and the colored circular magnets.

The instructions read to subjects appear in Appendix A.4. Subjects were instructed that the goal of this task was to identify clues that were related to one another and then signify any relationships between the groupings of clues using the dry erase markers or colored magnets (white, blue, green, yellow, red) and markers (red, green, black, blue). Subjects were then allowed 25 minutes to complete their analysis. Once complete, a photograph was taken of the diagram produced on the whiteboard.

Impoverished Condition

The impoverished condition provided neither the features to facilitate and encourage construction of a narrative or the visuospatial elements of the Narrative and Association conditions. Subjects were provided a Microsoft Excel spreadsheet that contained the clues in a randomized order. They were also given a Microsoft Word document that they could use to organize the clues and take notes. Subjects were allowed to use all of the features of Microsoft Excel and Word including copy and paste, sorting, and text formatting. Appendix A.4 contains the instructions read to the subjects. Once subjects had indicated that they understood the task, they were allowed 25 minutes to conduct their analysis.

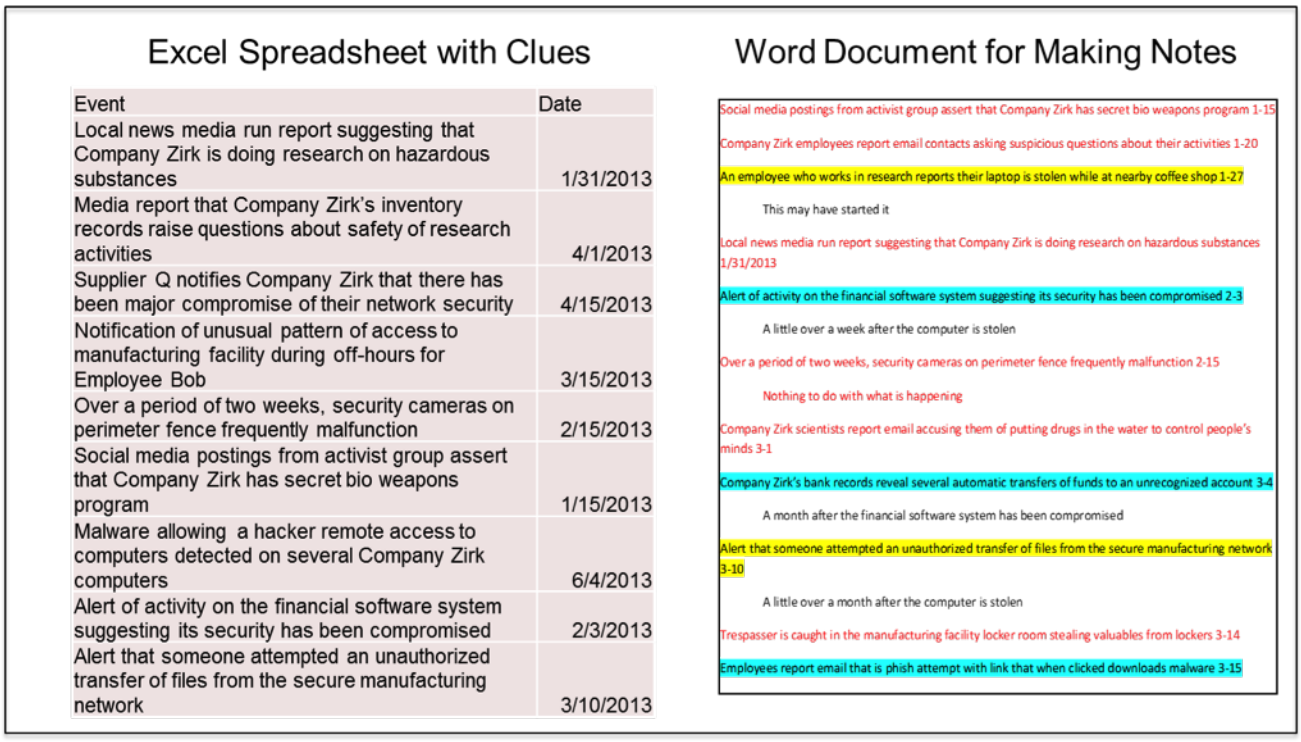


Figure 5.3: Example of Microsoft Excel spreadsheet with clues used in the Impoverished condition and the accompanying notes created by the subject using Microsoft Word.

Memory Recognition Test

The Narrative condition should have provided a basis for integrating clues into a meaningful structure that better incorporated the relationships between clues, with more elaborative processing, then the Association and Impoverished conditions. Consequently, subjects in the Narrative condition should have more robust memory representations of the clues. To test this hypothesis, a memory recognition test was administered. In the test, subjects were presented a series of clues and asked to indicate if each clues had appeared in the original set of clues. Each of the sixteen legitimate clues was presented. In addition, sixteen clues were presented that served as decoys. The decoys were constructed to resemble legitimate clues, yet varied with regard to critical details. For example, one of the legitimate clues stated, "Company Zirk employees report email contacts asking suspicious questions about their activities." This clue was altered to compose the corresponding decoy stating, "Company Z employees report email contacts requesting interviews to talk about their research." While similar, the legitimate clue implied that the contacts were suspicious, whereas the decoy offered no such implication.

For the memory recognition test, clues were presented for 5 sec, after which subjects were prompted to make a keyboard response to indicate if they believed that the clue had appeared in the original set. The instructions given to subjects specifically stated that for a clue to be considered to have appeared in the original set of clues, it must be identical in its description and wording.

Association Test

As previously stated, it was hypothesized that the Narrative condition would result in more robust memory representations of the clues. In addition to better recognition memory, this more robust representation should also result in stronger associations between the clues within given threads of the scenario for subjects with the Narrative condition. Thus, it was predicted that for pairs of clues within a given thread of the scenario, there would be stronger associations for subjects in the Narrative condition than there would be for subjects in the Association and Impoverished conditions. To test these predictions, pairs of clues were presented to the subjects and they were asked to indicate the extent to which each pair was related. For each pair, subjects rated the relatedness on a scale of 1 to 5, with 5 indicating that the clues were highly related to one another and 1 indicating that the clues are not related.

Event Reconstruction Test

The final measure asked subjects to provide their interpretation of the events using the software tool PlotWeaver. PlotWeaver provides a XML-based graphical interface for creating reconstructions of events. As shown in Figure 5.4, in diagramming stories, PlotWeaver diagramming stories allows entities and interactions between entities to be identified as a time-dependent series of events. For the Event Reconstruction Test, subjects were provided a brief tutorial by the experimenter showing how to use the key features of PlotWeaver. Features addressed in the tutorial included creating

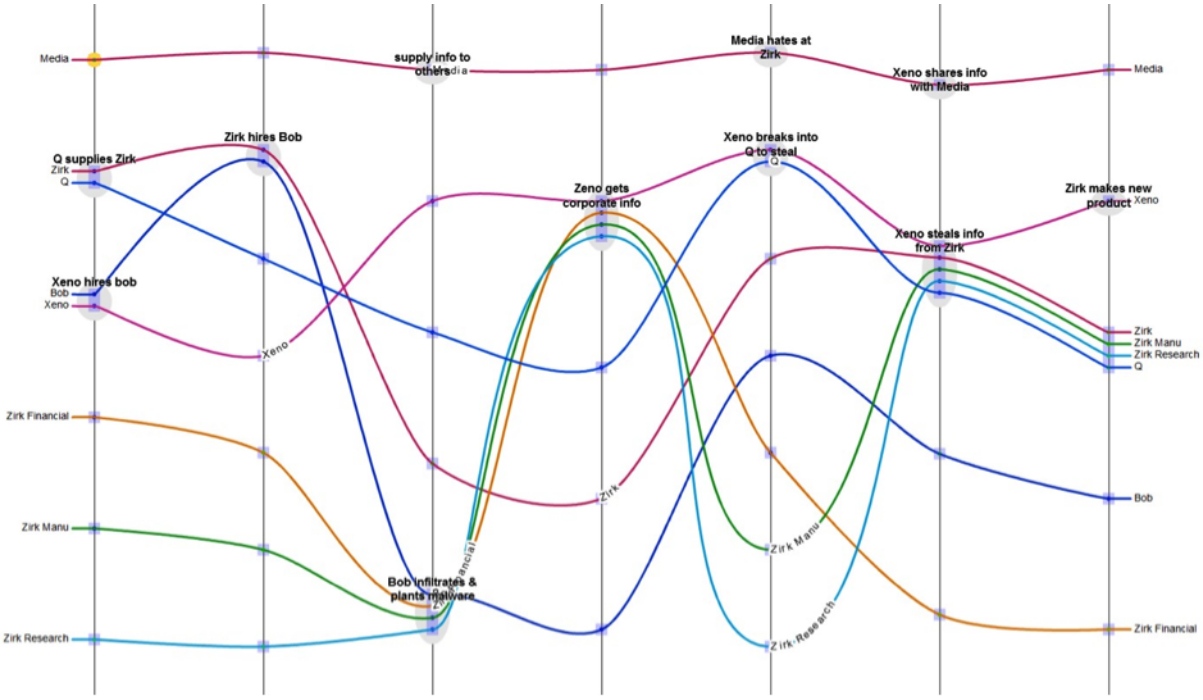


Figure 5.4: Example of a PlotWeaver diagram illustrating the subject's interpretation of events within the scenario.

story lines, adding time steps, merging story lines, splitting story lines, and inserting labels. Once the experimenter had verified that subjects understood how to use these features, they were given 25 min to create their PlotWeaver reconstruction of events. During this time, diagrams created by subjects in the Narrative and Association conditions and word documents created by subjects in the Impoverished condition were available and could be referenced at any time.

5.2 Results

5.2.1 Forensic Analysis Experience Survey

Figure 5.5 presents the average rating of experience for each activity involving criminal forensic analysis. While subject reported some experience with criminal forensic analysis in the context of literature, movies and television, and games, they had little experience in professional settings.

5.2.2 OSPAN

In Figure 5.6, the average score for the subjects in each experimental condition is presented for each measure obtained with the OSPAN. It may be observed that the average Absolute Score for

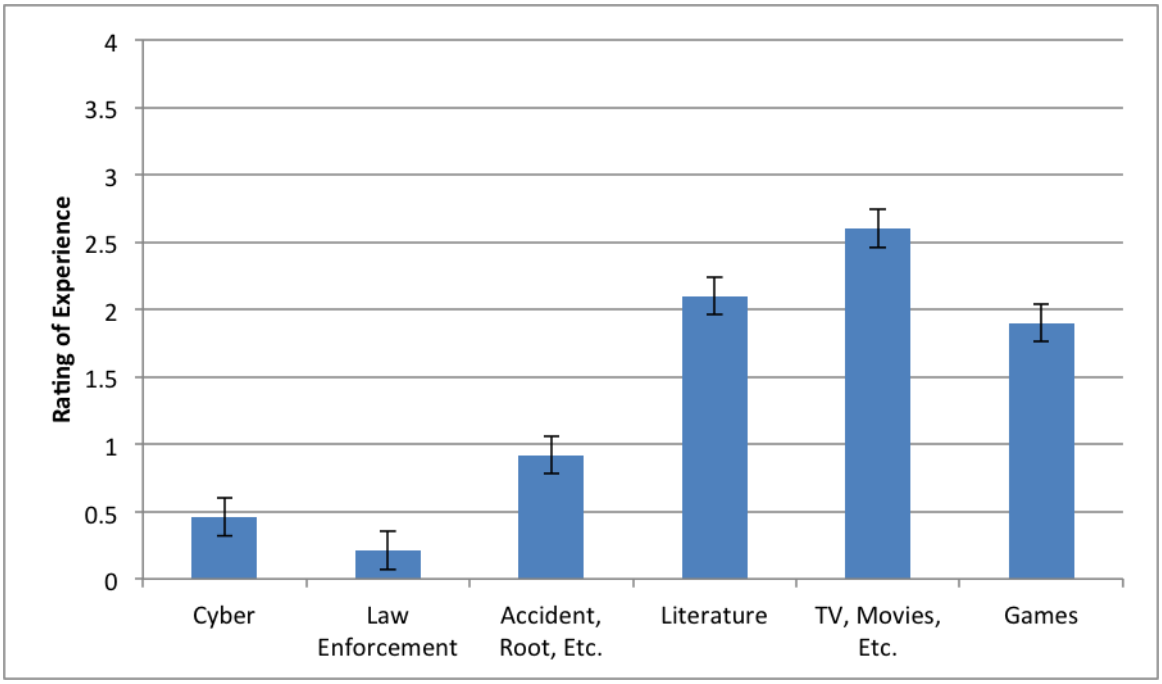


Figure 5.5: Self-reported experience with activities involving criminal forensic analysis (0="no experience;" 1="little experience;" 2="some experience;" 3="moderate experience;" and 4="extensive experience."

subjects in the narrative condition was noticeably greater than that for the other two conditions. A one-way ANOVA found that this difference was not statistically significant (F = 1.68 (df=2); NS).

5.2.3 Forensic Analysis

For subjects in the Narrative and Association conditions, an analysis was undertaken to assess which elements at the whiteboard to construct their diagrams. For this analysis, sixteen elements were identified which included the following:

- **Groups of Clues:** in affixing the laminated cards with the clues to the whiteboard, how many distinct groupings were formed? In calculating the total number of groupings, differing levels of groupings were each counted so that a low-level grouping of 2-3 clues was counted as one group and a higher-level grouping containing multiple low-level groupings was counted as another group.
- **Labels for Groups of Clues:** how many distinct labels for groupings were created using the dry erase markers.
- **Annotations:** how many instances were there in which subjects used the dry erase markers to create notes on the whiteboard, other than labels assigned to groups or links, or entries on

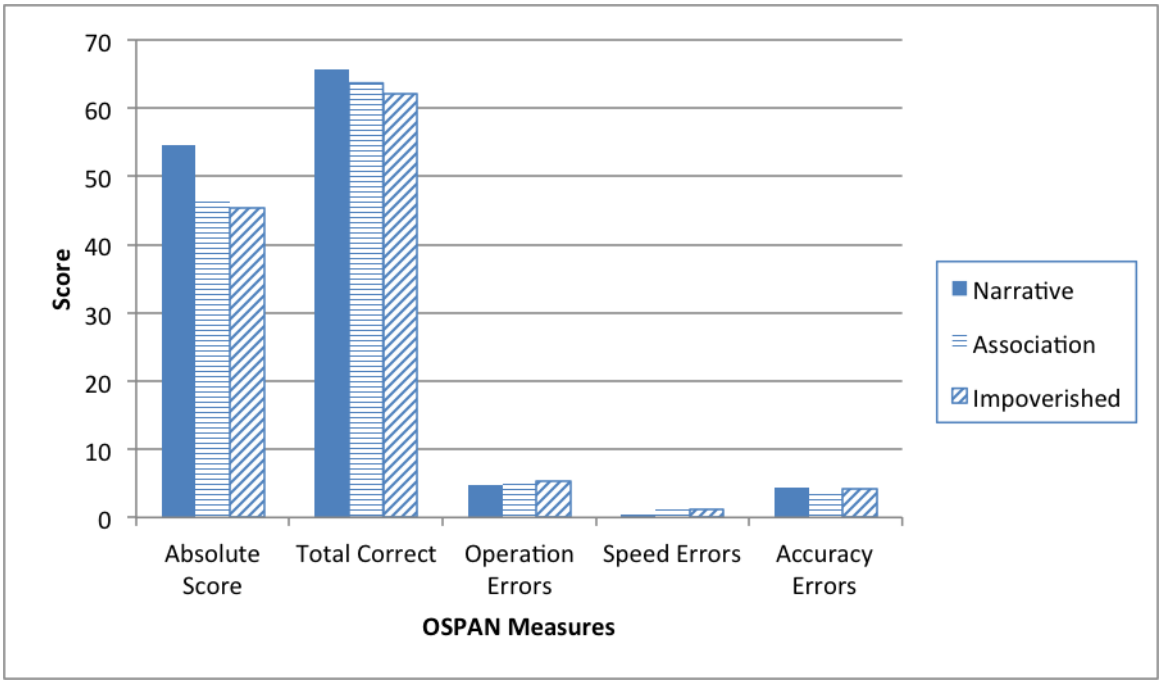


Figure 5.6: Average performance scores for each experimental condition for the performance measures produced by the OSPAN.

the Criminal Entity cards provided to subjects in the Narrative condition?

- **Links:** how many lines or arrows were drawn on the white board with the dry erase markers to signify that either clues or groups of clues were linked to one another?
- **Link Labels:** how many distinct labels for links were created using the dry erase markers?
- **Entities:** how many distinct entities were identified by either using the Criminal Entity cards provided to subjects in the Narrative condition or by noting entities using the dry erase markers?
- **Entity Motives:** how many instances were there in which entity motives were noted by making entries on the Criminal Entity cards provided subjects in the Narrative condition or by noting motives using the dry erase markers?
- **Tag Categories:** for subjects who used the circular colored magnets as tags, how many distinct categories of tags were used? Generally, this number corresponded to how many different colors of tags were used.
- **Tags Used:** how many of the circular colored magnets were used to tags?
- Questions – how many instances were there in which subjects denoted questions concerning either information that was unknown or uncertainty regarding their appraisal of events?

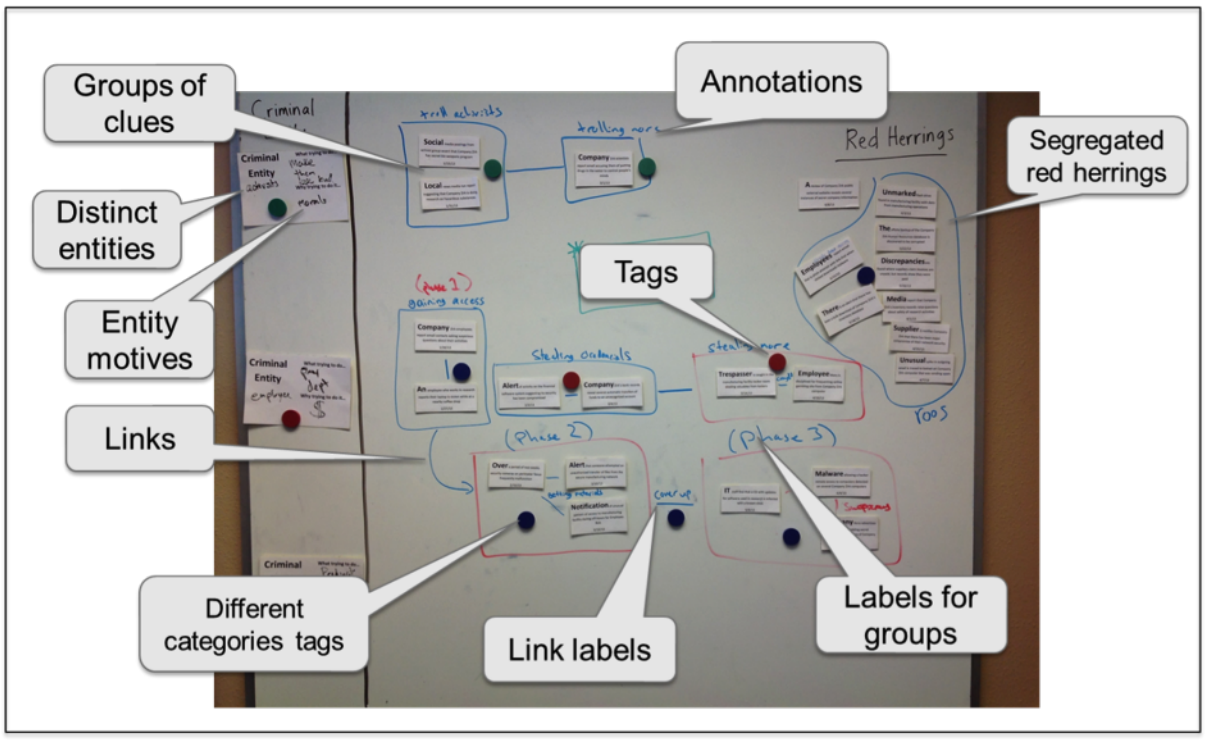


Figure 5.7: Examples of elements identified with whiteboard diagrams.

Figure 5.7 provides an illustration of each diagram element. Figure 5.8 shows the average frequency that elements were utilized by the subjects in the Narrative and Association conditions. It may be noted that the two groups differed little in the groups of clues, labels of groups of clues, annotations, links, labels of links and questions. However, subjects in the Narrative condition, who had been provided the Criminal Entity cards identified significantly more entities ($t=5.50$; $p < 0.001$) and entity motives ($t=7.48$; $p < 0.001$) than those in the Association condition. In contrast, on average, subjects in the Association condition used over twice as many tags, although this difference was not statistically significant ($t=1.25$; NS). This suggests that in interpreting events, subjects in the Association condition made as many conceptual distinctions as those in the Narrative condition. However, subjects in the Association condition did not have the Narrative framework furnished by the Criminal Entity cards and as a result, created more diverse conceptualizations.

There were two additional elements of the whiteboard diagrams that were not numerically quantified, but still considered. First, for each subject, it was determined whether in arranging clues, the subject created a chronological order. Each clue included a date on which the event was noted, although not necessarily the date the event occurred, that could be used to infer a chronological order of events. Chronological ordering of clues could occur in the Narrative condition by arranging clues in relation to the timeline that was provided. In either condition, chronological ordering could occur by arranging clues in order from earlier to later occurring events. As shown in Figure 5.9, subjects in the Narrative condition were significantly more likely to chronologically order clues than those in the Association condition ($t=3.99$; $p < 0.001$).

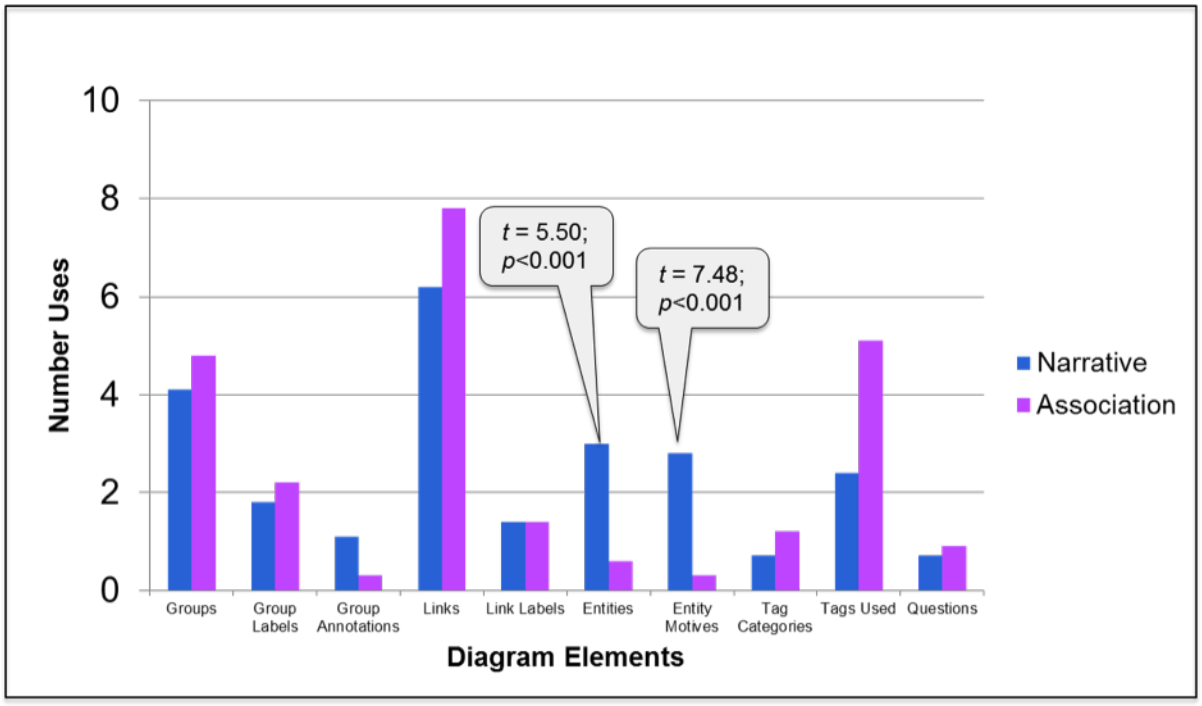


Figure 5.8: Use of different elements in constructing diagrams for the Narrative and Association conditions.

The second additional element concerned whether subjects segregated clues believed to be red herring clues from those believed to be legitimate clues. In the Narrative condition, this was determined on the basis of clues being placed in the upper right corner of the whiteboard in proximity to the "Red Herrings" label (i.e., Red Herrings Corner). In the Association condition the determination was more ambiguous and based on whether subjects had a group of clues that were separated from the other clues and had no lines or other demarcations indicating relationships between the clues or relationships to any of the other clues. Based on this determination, it was found that subjects in the Narrative condition were significantly more likely to segregate red herring from legitimate clues (See Figure 5.10).

5.2.4 Event Reconstruction Test

As discussed in an earlier section, the data suggests that subjects in the Narrative condition utilized the features provided to facilitate and encourage a narrative interpretation of events. This was evidenced by the subjects in the Narrative condition identifying more entities and entity motives, and exhibiting a greater tendency to order clues chronologically than subjects in the Association condition. Given that the Narrative condition had the intended influence on the analysis of the events, it may be asked what effect this had on their performance in the Event Reconstruction Test.

As shown in Figure 5.4, the PlotWeaver diagrams created for the Event Reconstruction Test

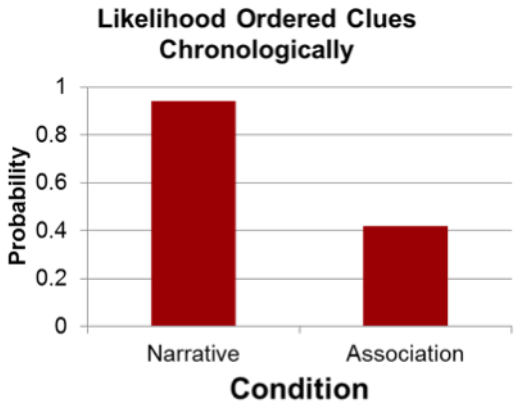


Figure 5.9: Probability subjects ordered clues chronologically.

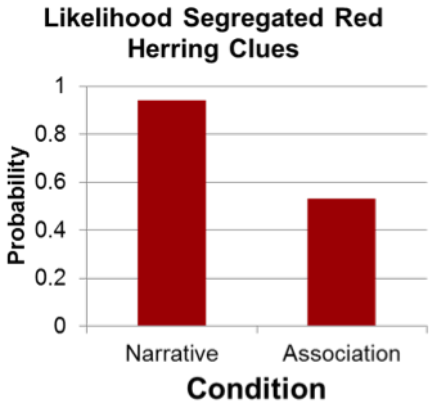


Figure 5.10: Probability subjects segregated red herring from legitimate clues.

contained a collection of story lines. In constructing these story lines, the individual nodes in the story lines generally coincided with specific clues. Analysis of the PlotWeaver diagrams occurred at three levels. Initially, there was a consideration of the clues appearing in the diagrams. It was found that overall, the subjects in the Narrative condition used more of the clues in their PlotWeaver diagrams ($F=3.49$ ($df=2$); $p < 0.05$). Notably, this difference corresponded to their using more of the legitimate clues ($F=3.37$ ($df=2$); $p < 0.05$), with there being little difference in their use of Red Herring clues ($F=0.55$ ($df=2$); NS) (See Figure 5.11).

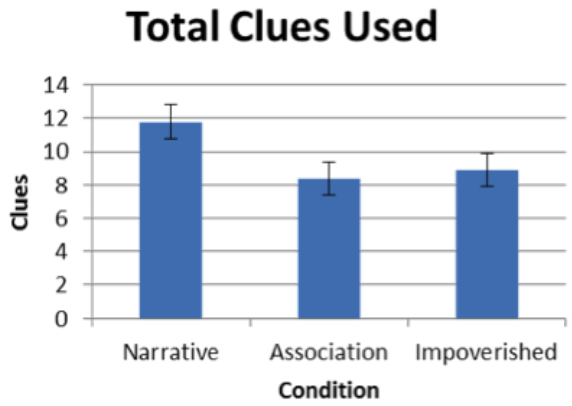
The second analysis of the PlotWeaver reconstructions considered the relationships between clues. If two clues appeared in the same PlotWeaver storyline, it was deemed that the subject believed that there was a relationship, or connection, between the clues. An analysis was undertaken that identified each instance in which subjects expressed a connection between a pair of clues based on them appearing within the same PlotWeaver storyline. It was found that while subjects in the Narrative condition identified more connections overall between pairs of clues and more connections between pairs of clues consisting of two legitimate clues, these differences were not statistically significant ($F=1.72$ ($df=2$); NS and $F=1.44$ ($df=2$); NS, respectively). Likewise, differences between experimental conditions for the number of connections between pairs of clues for which one or both clues was a Red Herring was not statistically significant ($F=1.63$ ($df=2$); NS). Figure 5.12 presents the results for connections between clues.

Finally, in comparing the connections identified between clues, there was consideration of the three threads. These connections would have involved instances in which a connection was identified between a pair of legitimate clues that were both elements of the same thread. There were 28 possible connections within the Hacktivist threat, and 6 each within the Criminal and Insider threads. While the subjects in the Narrative condition identified more connections within each of the three threads, there was a statistically significant difference for the Criminal thread ($F=5.68$ ($df=2$); $p < 0.01$), but not for the Hacktivist or Insider threads ($F=0.31$ ($df=2$); NS and $F=0.97$ ($df=2$); NS, respectively).

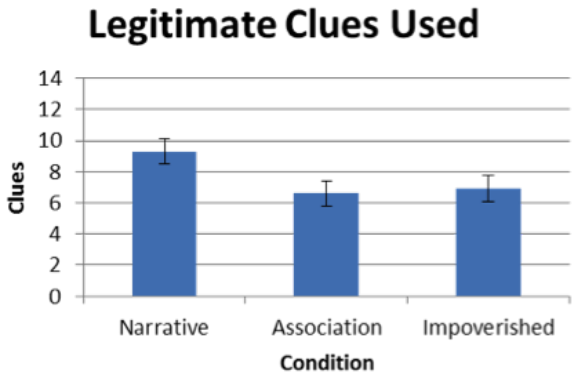
In scoring the PlotWeaver diagrams, there were many ambiguities due to there being an indirect mapping between the labels inserted onto the PlotWeaver diagrams by the subjects and the actual wording of the clues. To minimize inconsistencies in scoring from one subject to another, most of the plots (80%) were jointly scored by two experimenters. The remaining plots were separately scored by the same two experimenters with there being a 96% inter-rater reliability.

5.2.5 Relationship between Forensic Analysis and Event Reconstruction

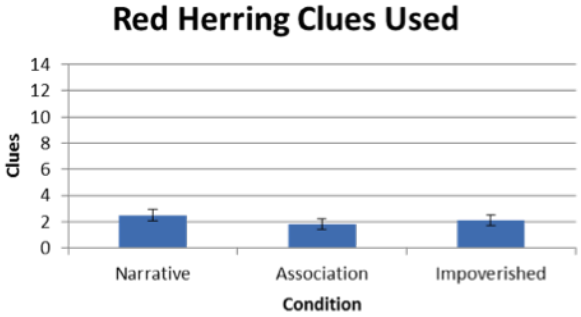
As previously discussed, an analysis of the elements used in constructing whiteboard diagrams during the forensic analysis found that subjects within the Narrative condition utilized the features meant to facilitate and encourage their developing a narrative account of events (e.g., Criminal Entity cards and timeline). It was next considered whether the use of any specific element(s) contributed to the performance of the subjects in the Narrative condition, and specifically, the findings that subjects within the Narrative condition incorporated more clues overall and more legitimate clues into their diagrams. A stepwise regression was performed for each of these dependent mea-



(a) b

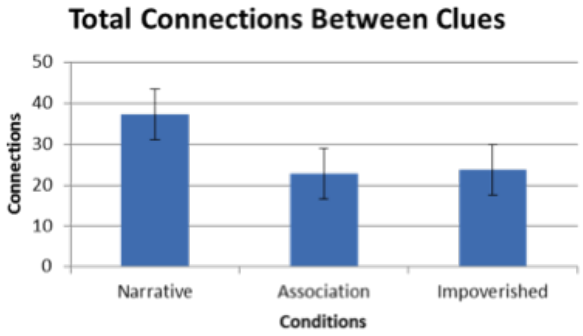


(b) b

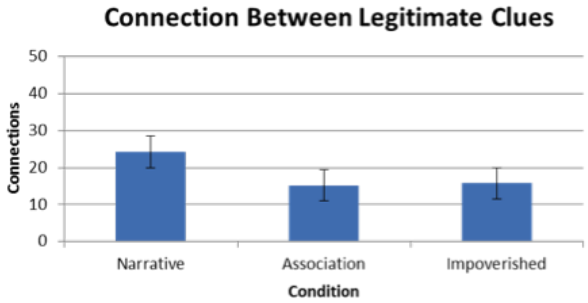


(c) b

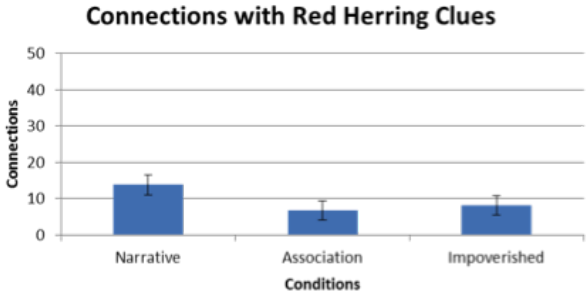
Figure 5.11: Subjects in the Narrative condition used more of the clues overall with this being a product of their using more of the legitimate clues, with all three groups incorporating approximately the same number of Red Herring clues.



(a) b



(b) b



(c) b

Figure 5.12: Comparison of experimental conditions for the number of connections identified between clues.

sures. Each of the twelve diagram elements discussed previously were assessed as predictors. A model was derived that accounted for a significant proportion of the variance in the number of clues used (R2=60.2; R2 adjusted=51.0, F=6.55; $p < 0.01$). This model incorporated three predictors variables listed in order of regression steps: Group Annotations (t=2.57; $p < 0.05$); Entities (t=1.72; $p < 0.01$); and Entity Motives (t=2.52; $p < 0.01$). Likewise, a model was derived that accounted for a significant portion of the variance in the number of legitimate clues used (R2=51.2; R2 adjusted=39.9, F=4.55; $p < 0.05$). This model incorporated the same three predictor variables listed in order of regression steps: Entities (t=3.13; $p < 0.01$; Group Annotations (t=1.99; $p < 0.10$); and Entity Motives (t=1.57; $p < 0.15$). None of the variables considered as possible predictors could be incorporated into a model that predicted a significant portion of the variance in the use of clues that were red herrings. Consequently, it is concluded that the superior performance of the subjects in the Narrative condition during the event reconstruction task may be attributable to the extent to which they used the Criminal Entity cards and annotated their diagrams.

Given that the subjects in the Association condition were not provided with the same elements to facilitate and encourage their development of a narrative account, there was interest in what elements might predict their performance. Stepwise regressions were calculated using the twelve diagram elements as predictors. These analyses failed to produce a model that accounted for a significant proportion of the variance for any of the performance variables. Thus, whereas the performance of subjects in the narrative condition was predicted on the basis of distinct whiteboard elements used in the forensic analysis, there were no comparable predictors that accounted for performance of subjects in the Association condition.

5.2.6 Relationship between other Predictors and Event Reconstruction Performance

Stepwise regressions were calculated to determine if any relationships existed between the self-reported experience of subjects for the items within the forensic experience survey and the measures of performance for the event reconstruction task. Self-reported experience on none of the items within the survey predicted a significant proportion of the variance in performance measures. Consequently, experience was concluded to not be a predictor of performance in the current study.

Similarly, stepwise regressions were conducted for the five measures of working memory capacities obtained from the OSPAN. None of the OSPAN measures predicted measures of performance for the event reconstruction task. Thus, working memory capacity was concluded to not be a predictor of performance for the event reconstruction task.

5.3 Conclusion

The current findings suggest that given an artifact that features elements encouraging users to construct a narrative account of events, users will employ these elements. Subsequently, performance measures associated with forensic event reconstruction are predicted by the extent to which subjects employ these elements. Finally, subjects given elements to facilitate their construction of a narrative perform better than those not provided such capabilities.

The current study has focused on the domain of cyber security forensic analysis. The results have direct bearing on the software tools provided to cyber security professionals, as well as cyber security education and training. There is currently an extremely lucrative market for software tools to support cyber security forensic analysis. While these software tools provide essential capabilities, generally, they do not offer utilities to translate the results of data analysis (e.g., packet capture analysis) into a meaningful narrative. Consequently, as has been previously reported, cyber security professional frequently turn to additional artifacts (e.g., Excel spreadsheets, digital notepads) to facilitate their analysis [27], with performance predicted on the basis of the extent to which individuals utilize these supporting artifacts [26]. While discussed here in the context of cyber security forensic analysis, it may be inferred that the same conclusions apply to other domains that involve the reconstruction of series of events (e.g., law enforcement and medical forensic analysis, accident and root cause analysis, etc.)

A key finding from the current study concerns the need to facilitate the identification of entities and, their intents and motives. While subjects utilized the timeline provided in the Narrative condition, use of the timeline was not predictive of performance. The current study did not provide an explicit mechanism for representing the location of events, or other representations of contexts, it was observed that many subjects created their own spatial references. For example, in the current study, each of the three threads involved a somewhat distinct component of the overall computer network of the fictitious company. The Hacktivist thread primarily involved engineering and research resources, the Criminal thread exclusively concerned the financial systems and the Insider thread exclusively focused on manufacturing resources. It was observed that many of the subjects used annotations to capture these distinctions, with components of the computing network corresponding to distinct spatial references. Likewise, it was noted that the erroneous connections made by subjects often involved their failure to recognize distinctions between these spatial references. This suggests that while it was not practical with the current physical instantiation (i.e., whiteboard with laminated magnetic cards and differently colored magnets), users may benefit from features that additionally facilitate their incorporating spatial references into their narrative accounts.

Chapter 6

Conclusion

In this report we have discussed research performed under the Nested Narratives LDRD. Our work followed two main lines of inquiry. First, we developed LAASER-ttag, a toolkit for instrumenting distributed systems such as Hadoop clusters to follow the movement of pieces of data. Our goal here was to construct a chain of causal attribution from activity on a target system back to a process (and ultimately user activity) on some other system. We learned along the way that even synchronized system-call transcripts across an entire testbed do not resolve the underlying causes of ambiguity in attribution. This suggests that a correlative approach is more suited than one based on strict causality.

Second, we tested our hypothesis that tools supporting narrative formation lead to better performance when extracting and explaining events from a collection of clues. We fabricated a plausible cybersecurity forensics task and evaluated the performance of 52 employees of Sandia National Laboratories under one of three separate conditions. Each condition provided a different level of narrative support. Our results support our hypothesis.

Our plans to develop prototype tools for cybersecurity forensic analysis based on real data and real analyses had to be set aside when security concerns overruled our request for access to said data.

While this LDRD project centered upon the use of narrative in cybersecurity, we believe that the nested representation of multiscale narrative has far broader applicability. In our future work we will interact with different groups in different mission areas at Sandia to bring robust narrative support to a wide range of domains.

References

[1] LTTng: Filling the gap between kernel instrumentation and a widely usable kernel tracer.

[2] Laura Chappell and Gerald Combs. *Wireshark 101: Essentiall skills for network analysis*. Laura Chappell University, 2013.

[3] A. R. Conway, N. Cowan, M. F. Bunting, D. J. Therriault, and S. R. Minkoff. A latent variable analysis of working memory capacity, short-term memory capacity, processing speed, and general fluid intelligence. *Intelligence*, 30:163 – 183, 2002.

[4] National Research Council. Strengthing forensic science in the United States: a path forward, 2009.

[5] Sean Crosby, Nick Pattengale, Craig Ulmer, and Vince Urias. Nephelae LDRD project summary. Technical Report SAND2012-8807, Sandia National Laboratories, 2012.

[6] Christopher E. Davis, Lon A. Dawson, Arlo L. Ames, Theodore M. Reed, Samuel D. Olsen, Michael G. Stickland, David R. Grochocki, Nicholas D. Pattengale, Anna M. Larez, and Brian P. Van Leeuwen. Cyber operations research and network analysis (CORONA) year-1 final report. Technical Report SAND2012-5633, Sandia National Laboratories, 2012.

[7] Andrzej Duda, Gilbert Harrus, Yoram Haddad, and Guy Bernard. Estimating global time in distributed systems. In *ICDCS*, pages 299–306, 1987.

[8] GlusterFS Project. GlusterFS. `http://www.gluster.org`.

[9] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable parallel programming with the Message-Passing Interface*. The MIT Press, 3 edition, 2014.

[10] Shelby Hopkins, Andrew Wilson, Austin Silva, and Chris Forsythe. Facilitation of forensic analysis using a narrative template. In *Proceedings of HCI International*, 2015.

[11] Shelby Hopkins, Andrew Wilson, Austin Silva, and Chris Forsythe. Factors contributing to performance for cyber security forensic analysis. In *Applied Human Factors and Ergonomics 2015*, 2015.

[12] Ibm i2 analyst's notebook. `http://www-03.ibm.com/software/products/en/analysts-notebook`. Accessed 5 January 2015.

[13] Nancy Iskander, Matthew Thorne, and Craig Kaplan. Comic book narrative charts. `http://csclub.uwaterloo.ca/~n2iskand/?page_id=13`. Accessed 5 January 2015.

[14] Bart Jacob, Paul Larson, Breno Henrique Leitao, and Sulo Augosto M Martins da Silva. SystemTap: Instrumenting the Linux kernel for analyzing performance and functional problems. Technical Report REDP-4469-00, IBM, 2009.

[15] S. M. Kassin, I. E. Dror, and J. Kukucka. The forensic confirmation bias: Problems, perspectives, and proposed solutions. *Journal of Applied Research in Memory and Cognition*, 2:42 – 52, 2013.

[16] Andy Konwinski and Matei Zaharia. Finding the elephant in the data center: Tracing Hadoop. Technical Report CS294, University of California, Berkeley, 2008.

[17] LTTng Project. Linux Tracing Toolkit, Next Generation. `http://lttng.org`.

[18] Randall Munroe. XKCD #657: Movie Charts. `http://xkcd.com/657/`. Accessed: 2015-01-05.

[19] Michael G. Noll. Running Hadoop on Ubuntu Linux (multi-node cluster). `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/`.

[20] Michael Ogawa and Kwan-Liu Ma. Software evolution storylines. In *Proceedings of the 5th International Symposium on Software Visualization*, SOFTVIS '10, pages 35–42, New York, NY, USA, 2010. ACM.

[21] OpenStack Group. OpenStack. `http://openstack.org`.

[22] Plotweaver: Automating xkcd's movie character interaction graphs. `http://infosthetics.com/archives/2010/06/plotweaver_automating_xkcds_movie_character_interaction_graph.html`. Accessed 5 January 2015.

[23] Benjamin Poirier, Robert Roy, and Michel Dagenais. Accurate offline synchronization of distributed traces using kernel-level events. *SIGOPS Oper. Syst. Rev.*, 44(3):75–87, August 2010.

[24] Dave Shreiner, Graham Sellers, John Kessenich, and Bill Licea-Kane. *OpenGL Programming Guide*. Addison-Wesley Professional, 8 edition, 2013.

[25] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[26] A. Silva, J. McClain, T. Reed, B. Anderson, K. Nauer, R. Abbott, and C. Forsythe. Factors impacting performance in competitive cyber exercises. In *Proceedings of the Interservice/Interagency Training Simulation and Education Conference*, 2015.

[27] A. Singh, L. Bradel, A. Endert, R. Kincaid, C. Andrews, and C. North. Supporting the cyber analytic process using visual history on large displays. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, 2011.

[28] Unified Modeling Language specification. `http://www.uml.org`. Accessed: 2015-01-05.

[29] N. Unsworth, R. P. Heitz, J. C. Schrock, and R. W. Engle. An automated version of the operation span task. *Behavior research methods*, 37:498–505, 2005.

[30] Vincent E. Urias and Munawar Merza. Splunking the cloud. Presented at Splunk User Conference, 2011.

[31] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: a scalable, high-performance distributed file system. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, pages 22–22, Berkeley, CA, USA, 2006. USENIX Association.

[32] R. A. Zwaan and G. A. Radvansky. Situation models in language comprehension and memory. *Psychological Bulletin*, 123:162, 1998.

Appendix A

User Study Instructions

A.1 Pretense

The following was read to subjects to provide background prior to their commencing with the forensic analysis.

"You have been asked to look into a collection of suspicious events at Company Zirk. This company is a highly successful developer and manufacturer of vaccines distributed across many developing nations. You will be provided a collection of clues to one or more crimes that have been committed. Your task is to analyze the clues and determine what happened. Be aware, that some of the clues are red herrings and do not relate to the events you are investigating."

A.2 Crime Scenarios

The following describes each of the criminal threads that made up the overall scenario. The bolded text corresponds to the clues that were made available to subjects for the forensic analysis.

A.2.1 Hacktivist Thread

A Hacktivist group suspects Company Zirk's vaccine business is actually a cover for a secret government bioweapons program. Their objective is to expose Zirk. They post their suspicions on social media and contact Zirk employees to ask about their work. They also hang out at a local coffee shop that Zirk employees frequent hoping to overhear conversations. An employee working in research leaves his laptop unattended and it is stolen by a Hacktivist group member. The group finds various files on the computer regarding research activities at Zirk and contacts the media claiming they have proof that Zirk is developing biological weapons. The media is unwilling to report these claims, but instead, the media reports that there is evidence of hazardous operations. The Hacktivists decide that they must get onto the computer systems at Zirk to find the evidence they need to support their claims. Their next step is to send Zirk employees a phishing email disguised to be from a contractor who provides IT support. The phish claims that the annual license

for Microsoft Office is about to expire and they must click the accompanying link to renew. Several employees click the link which downloads malware onto employees' computers that provides a back door for the Hacktivists to remotely access their machines. While the hacktivists are unable to access the research or manufacturing networks, they do find the inventory database and perform a bulk download of its contents. Based on this information, they return to the media and repeat their claims asserting that Company Zirk stocks all the materials they would need to create bioweapons. Instead reporting these claims, the media run a report about the safety of Zirk operations.

A.2.2 Criminal Thread

Through various mechanisms, a criminal organization has thoroughly compromised the computer network at Supplier Q, which is a major supplier to Zirk. The criminal organization sees that Supplier Q does business with Zirk and realizes that Zirk is a more lucrative target. All of the purchase orders and invoices between Zirk and Supplier Q are done electronically. Malware is attached to an electronic invoice that allows the criminal organization to get a foothold on the financial system at Zirk. The malware sets off an alert, but only after the criminal organization's hackers have inserted multiple back doors to Zirk's financial system. When Zirk financial staff approve an invoice from a supplier, an electronic transaction is sent to the bank requesting funds be transferred from Zirk's account to the account of the supplier. The criminal organization installs malware that intercepts these transactions and alters the data fields so that funds are instead transferred into a bank account the criminal group controls. Zirk financial staff recognizes that funds have been transferred into an unrecognized account and soon thereafter, suppliers begin to alert Zirk that their invoices have gone unpaid.

A.2.3 Insider Thread

An employee for Company Zirk, Bob, is leaving the company to take a job with a competing company, Xeno. Zirk has a revolutionary manufacturing process and Bob knows that he will become a favorite at his new job if he knows how to reproduce Zirk's manufacturing capabilities. The manufacturing process is instantiated within the Numeric Control programs used to drive the machinery used in manufacturing the vaccines. Bob's objective is to acquire these programs and the data generated from several manufacturing runs. Bob is not a very good Insider and first tries to send files to an off-site computer, but the firewall blocks this attempt. Since this did not work, he decides he'll do it the hard way and transfer the information to flash drives while no one is around. However, he gets sloppy and leaves one of the flash drives behind. In the process, the access control system for the manufacturing facility detects and issues an alert concerning Bob's entering and leaving at odd-hours. Bob does get enough information that he is able to help Xeno replicate the manufacturing processes of Zirk, which is soon advertised by Xeno at a trade show attended by Zirk personnel.

A.3 Clues

A.3.1 Legitimate Clues

The following clues were provided to subjects for the forensic analysis.

Hacktivists Thread

- Social media postings from activist group assert that Company Zirk has secret bio weapons program (1/15/13)
- Company Zirk employees report email contacts asking suspicious questions about their activities (1/20/13)
- An employee who works in research reports their laptop is stolen while at nearby coffee shop (1/27/13)
- Local news media run report suggesting that Company Zirk is doing research on hazardous substances (1/31/13)
- Employees report email that is phish attempt with link that when clicked downloads malware (3/15/13)
- There is an alert that there has been a bulk download of Company Zirk's inventory database (3/18/13)
- Media report that Company Zirk's inventory records raise questions about safety of research activities (4/1/13)
- Malware allowing a hacker remote access to computers detected on several Company Zirk computers (6/4/13)

Criminal Thread

- Alert of activity on the financial software system suggesting its security has been compromised (2/3/13)
- Company Zirk's bank records reveal several automatic transfers of funds to an unrecognized account (3/4/13)
- Supplier Q notifies Company Zirk that there has been major compromise of their network security (4/15/13)
- Discrepancies are found where suppliers claim invoices are unpaid, but records show they were paid (5/26/13)

Insider

- Alert that someone attempted an unauthorized transfer of files from the secure manufacturing network (3/10/13)
- Notification of unusual pattern of access to manufacturing facility during off-hours for Employee Bob (3/15/13)
- Unmarked flash drive found in manufacturing facility with data from manufacturing operations (4/3/13)
- Company Xeno advertises capability resembling secret manufacturing processes of Company Zirk (6/15/13)

A.3.2 Red Herrings

The following clues were provided to subjects in addition to the legitimate clues and served as red herrings for the forensic analysis.

- Over a period of two weeks, security cameras on perimeter fence frequently malfunction
- Company Zirk scientists report email accusing them of putting drugs in the water to control people's minds
- Trespasser is caught in the manufacturing facility locker room stealing valuables from lockers
- Unusual spike in outgoing email is traced to botnet on Company Zirk computer that was sending spam
- Employee Mary is disciplined for frequenting online gambling site from Company Zirk computer
- IT staff find that a CD with updates for software used in research is infected with a known virus
- The offsite backup of the Company Zirk Human Resources database is discovered to be corrupted
- A review of Company Zirk public external website reveals several instances of secret company information

A.4 Instructions

The following instructions were read to subjects for each forensic analysis experimental condition.

Narrative Condition Instructions

"Your clues each appear on a separate card with each card providing a description of an event and the date(s) that the event occurred. You will be given a total of 25 minutes to analyze, organize, and identify relationships between the clues. To help organize the clues, please use the whiteboard and labeled columns. You should attach the clues to the whiteboard and use the markers to identify relationships between the clues."

Association Condition Instructions

"Your clues each appear on a separate card with each card providing a description of an event and the date(s) that the event occurred. You will be given a total of 25 minutes to analyze, organize, and identify relationships between the clues. To help organize the clues, please use the whiteboard. You should place similar or related clues closer together and less related clues farther apart on the board. Also, you should use the markers to identify relationships between the clues."

Impoverished Condition Instructions

"Your clues each appear as an entry in a database with each entry providing a description of an event and the date(s) that the event occurred. You will be given a total of 25 minutes to analyze, organize, and identify relationships between the clues. To help organize the clues, please use the Microsoft Word document that is provided. You should copy and paste entries into the Word document, adding your own notes to identify relationships between the clues."

DISTRIBUTION:

1	MS 0359	D. Chavez, LDRD Office, 1911
1	MS 0813	Beth Potts, 9312
1	MS 1326	Andrew Wilson, 1461
1	MS 0899	Technical Library, 9536 (electronic copy)


Sandia National Laboratories